Bachelorarbeit

Philipp Baumgart

Ansteuerung, Regelung und Erprobung eines Servomotors für einen Wellenkanalantrieb

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Philipp Baumgart

Ansteuerung, Regelung und Erprobung eines Servomotors für einen Wellenkanalantrieb

Bachelorarbeit eingereicht im Rahmen der Bachelor-/Masterprüfung

im Studiengang Maschinenbau Energie- und Anlagensysteme
am Department Maschinenbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
HAW Hamburg
La Ola Wellenkanal
Berliner Tor 5
20099 Hamburg

Erstprüfer : Prof. Dr.-Ing. Peter Wulf
Zweitprüfer : Prof. Dr.-Ing. Stefan Wiesemann

Abgabedatum: 18.08.2015

Zusammenfassung

Name des Studenten
Philipp Baumgart

Thema der Bachelorthesis
Ansteuerung, Erprobung und Regelung eines Servomotors für einen Wellenkanalantrieb

Stichworte
Steuerung, Wellenkanal, LabVIEW, LaOla, Wellenkraftwerk

Kurzzusammenfassung
Diese Arbeit beschäftigt sich mit der Ansteuerung und Erprobung eines Servomotors für einen Wellenkanalantrieb mittels LabVIEW. Zu diesem Zweck wurde eine TCP Verbindung des Rechners mit dem Frequenzumrichter gewählt und ein Programm mit LabVIEW geschrieben, welches die Steuerung realisieren soll.
Des Weiteren wird auf die Möglichkeit einer Steuerung mittels Modbus/TCP eingegangen und die einzelnen Schritte zur Programmierung der TCP Verbindung erläutert. Außerdem werden die technischen Grundlagen von Netzwerken im Rahmen der Arbeit erläutert um ein eingehendes Verständnis der Theorie zu gewährleisten.

Name of Student
Philipp Baumgart

Title of the paper
Controlling, testing and regulation of a servomotor for a wavecanal

Keywords
Controlling, wave canal, LabVIEW, LaOla, wave power plant

Abstract
The following bachelor thesis deals with controlling and testing of a servomotor for a wave cannel drive via LabVIEW. For this purpose a TCP connection between the computer and the frequency inverter was created and a program with LabVIEW was written in order to achieve control of the servomotor.
Furthermore, the thesis adresses the possibility of controlling the servomotor via Modbus/TCP, as well as the necessary steps required to program the TCP connection.
In addition, the technical fundamentals of networks as part of the thesis are explained thoroughly in order to ensure comprehension of the relevant theory.

Aufgabenstellung

Im Zentrum für Energietechnik an der HAW Hamburg soll im Rahmen des studentischen Projekts LaOla ein neuer Wellenkanal in Betrieb genommen werden. Der Wellenkanal besteht aus einem 10m langen Tank mit transparenten Wänden. Am Wellenkanal sollen zukünftig Prototypen von Wellenkraftwerken im Labormaßstab untersucht werden.

Die Wellen sollen durch einen Wellenerzeuger generiert werden, bei dem eine Umschaltung zwischen Schwing- und Hubbewegung möglich ist. Für die aktuell laufende Inbetriebnahmephase steht dabei jedoch zunächst die Schwingbewegung im Vordergrund. Der Wellenerzeuger soll in den späteren Untersuchungen durch die Überlagerung von einzelnen phasenverschobenen Wellen mit jeweils eigener Amplitude und Frequenz das vorgegebene Seegangsspektrum eines irregulären Seegangs möglichst realistisch abbilden. Um dies realisieren zu können, ist es notwendig, den Elektromotor des Wellenerzeugers zuverlässig und in Quasi-Echtzeit über eine noch zu definierende Schnittstelle aus der LabVIEW-Umgebung zuverlässig an steuern und in ein Regelkonzept einbinden zu können.

Im Einzelnen sind dazu folgende Aufgaben im Rahmen dieser Bachelorthesis vorgesehen:

- Einarbeitung in die vorliegende Antriebstechnik („SEW-Syncron-Servomotor“), die vorhandenen Schnittstellen und in die LabVIEW-Umgebung
- Auswahl einer geeigneten Motor-Schnittstelle und Herstellung einer zuverlässigen Verbindung zwischen LabVIEW und dem Antriebssystem
- Definition, Auswahl und Programmierung von einfachen Testsequenzen in LabVIEW (z.B. Winkelverstellung, Verfahrgeschwindigkeit, …) und Test am realen Motor
- Einbindung und Test der ausgewählten Schnittstelle in ein LabVIEW-Programm zur Regelung des Wellenkanalantriebs (aus [6])
- Einbindung und Test der Schnittstellenkarte eines Wellendrahtmesssystems (Ist-Wert-Geber) in das LabVIEW-Programm (s. [3])
- Ausführliche Erprobung der gesamten Steuerungs- bzw. Regelstrecke am realen Motor
- Ausführliche Dokumentation der Schnittstellen, der Erprobungen und der erzielten Ergebnisse

Bei den praktischen Tests ist sicherzustellen, dass der Servomotor nicht mit der Mechanik des sog. „Paddels“ verbunden ist und frei drehen kann. Damit soll sichergestellt werden, dass bei einem ungünstig verlaufenden Test keine Schäden an Personen oder Anlagen auftreten. Falls ferner elektrische oder mechanische Arbeiten am Antrieb notwendig werden, dürfen diese nur von dafür autorisierten Personen der HAW Hamburg durchgeführt werden.

Der Arbeit ist ein Datenträger (CD oder DVD) mit der Arbeit selbst (im PDF-Format) sowie allen relevanten Dateien (insbesondere den LabVIEW-Dateien und ggf. verwendeten Treibern für die Schnittstelle) beizulegen. Die Leitlinien des Instituts ICAMM zur formalen Gestaltung der Arbeit sind zu beachten.

Zu beachtende Vorarbeiten:

[1] P. Baumgart, B. Niemax: CAN-Bus-Ansteuerung für einen elektrischen Wellenkanalantrieb. Hausarbeit HAW Hamburg, 2014.

[2] M. Hartlev, G. Nobis: Inbetriebnahme und Erprobung einer Steuerung für einen Wellenkanalantrieb. Hausarbeit HAW Hamburg, 2014.

[3] A. Klemichen, M. Simon: Inbetriebnahme, Eichung und Erprobung eines Wellendrahtmesssystems für einen Wellenkanal. Masterprojekt HAW Hamburg, 2014.

[4] L. Assfalg, S. Decher, D. Karathomas, H. Krieg, R. Templin: Wellenkanalantrieb - Optimierung einer vorhandenen Konstruktion und Bau des Prototyps. Bachelorprojekt, HAW Hamburg, 2013.

[5] M. Kirk, J. Pless, H. Röhrs, F. Schulz: Redesign und Konstruktion eines Wellenerzeugers für den HAWWellenkanal. Bachelorprojekt, HAW Hamburg, 2012.

[6] N. Krieger: Entwicklung einer Regelung für den Antrieb einer maritimen Versuchsanlage. Bachelorthesis, HAW Hamburg, 2012.

Abkürzungs- und Formelverzeichnis

EEG	Erneuerbare-Energien-Gesetz
EMEC	Europe Marine Energy Centre
HAW	Hochschule für angewandte Wissenschaften Hamburg
TCP	Transmission Control Protocol
IP	Internetprotocol
OSI	Open Systems Interconnection
TCP/IP	Transmission Control Protocol / Internet Protocol
EtherNet/IP	EtherNet Industrial Protocol
E/A	Eingang/Ausgang
SYN	synchronize Massage
ACK	acknowledgement
IANA	Internet Assigned Numbers Authority
ICANN	Internet Corporation for Assigend Names and Numbers
UDP	User Datagram Protocol
CIP	Common Industrial Protocol
SMLP	Simple Movilink Protocol
LRC	longitudinal redundancy check
RTU	Remote Terminal Unit
CRC	cyclic redundancy check
IRC	International Electotechnical Commission
VI	Virtual Instrument
ms	Millisekunde

ω	Winkelgeschwindigkeit	[rad/s]
α	Winkel	[rad]
n	Drehzahl	[1/min]
t	Zeit	[sek]

Abbildungsverzeichnis

Tabellenverzeichnis

Inhaltsverzeichnis

1. Einleitung

1.1 Gesamtkontext

Durch die Energiewende in Deutschland, die durch den Atomausstieg vom 14. März 2011 verschärft wurde ist es vermehrt nötig in Deutschland erneuerbare Energien zu fördern und diese massiv auszubauen. Zum 6. August 2011 verloren acht deutsche Atomkraftwerke ihre Betriebserlaubnis, dies wurde in der Bundeskabinettssitzung vom 6. Juni 2011 beschlossen [1][2]. Bis zum Jahr 2022 sollen alle deutschen Atomkraftwerke abgeschaltet werden. Die hieraus resultierende Differenz in der Energieerzeugung muss durch neue Kraftwerke ausgeglichen werden. Im Rahmen des Gesetzes für den Ausbau erneuerbarer Energien (Erneuerbare-Energien-Gesetz EEG 2014) soll der Anteil der erneuerbaren Energien bis zum Jahr 2025 auf 40-45 Prozent am Bruttostromverbrauch steigen [3], hierdurch können die Verluste der Atomenergie behoben werden. Im Zuge des Gesetzes sollen vor allem Windenergie und solare Strahlungsenergie gefördert und ausgebaut werden [4]. Das Gesetz als solches berücksichtigt die Wasserkraft nur minimal, da die konventionelle Wasserkraft in Deutschland bereits nahezu vollständig erschlossen ist. Der Betrag der konventionellen Wasserkraftwerke an der gesamten Energieerzeugung betrug zum Stand Dezember 2013 3,4% [5] (Abbildung 1).

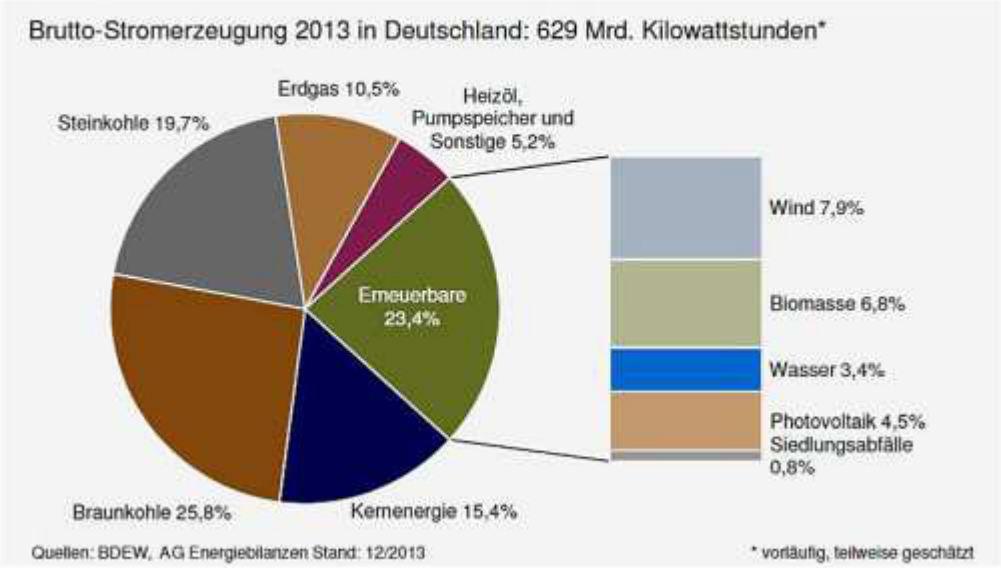

Abbildung 1: Brutto-Stromerzeugung 2013 [5]

Um die Wasserkraft weiter auszubauen und somit eine im Gegensatz zu Windenergie und Solarenergie stetige und Wetter unabhängige Energiegewinnung zu gewährleisten ist es nötig neue Wege zu gehen. Zu diesem Zweck muss die Entwicklung von neuartigen Wasserkraftanlagen, wie Wellenhubkraftwerken und Strömungs- beziehungsweise Gezeitenkraftwerken voran getrieben werden.

Ein erster Schritt in dieser Entwicklung ist das vom Europe Marine Energy Centre (EMEC) getestete und von den Firmen Pelamis Wave Power und Enersis entwickelte Seeschlangenkraftwerk (Pelamis) [6], welches vor der schottischen Küste erprobt wurde.

Abbildung 2: Pelamis (Seeschlange) im EMEC Test vor Schottland [6]

Des Weiteren werden küstengebundene Wellenkraftwerke, die mittels Wellenhub und dem damit erzeugten Luftstrom durch eine Wells-Turbine Strom erzeugen, getestet. Wells-Turbinen drehen sich unabhängig der Anströmrichtung nur in eine Richtung, was Energieverluste durch den Wechsel der Drehrichtung verhindert. Die Firma Voith baute eine solche Anlage im November des Jahres 2000 an der Küste der schottischen Insel Islay.

Das Potential der Wellen kann jedoch auf hoher See besser genutzt werden, zu diesem Zweck müssen neue Prototypen zunächst in Simulationen und maßstäblichen Versuchen erprobt werden. Hierzu wurde an der Hochschule für angewandte Wissenschaften Hamburg (HAW) das Projekt LaOla ins Leben gerufen. Im Laufe des Projekts wurde ein Wellenkanal mit den Ausmaßen 10m x 1,50m x 1m (LxBxH) (Abbildung 3) zum Test von Modellen gebaut.

Abbildung 3: Wellenkanal [7]

Die Anlage soll nach der Fertigstellung und Inbetriebnahme im Laborbetrieb eingesetzt werden und als Forschungsprojekt geführt werden.

1.2 Zielsetzung

Ziel dieser Arbeit ist es eine Möglichkeit für die Steuerung des Stellmotors, welcher das Wellenpaddel zur Wellenerzeugung antreibt, zu realisieren. Dies soll mit einer Programmierung durch LabVIEW und einer ausgewählten Schnittstelle erfolgen.

Die möglichen Schnittstellen werden durch die Anschlussmöglichkeiten des Frequenzumrichters festgelegt. Der DFE33B verfügt über Anschlüsse für Modbus (X30, X32), CANOpen (X15) und EtherCat (X30, X32), diese sind in Abbildung 4 zu sehen. Da sich die Hausarbeit CAN-Bus-Ansteuerung für einen elektrischen Wellenkanalantrieb bereits mit der Möglichkeit einer LabVIEW Steuerung mittels CANOpen beschäftigt hat, wird diese Schnittstelle nicht weiter betrachtet.

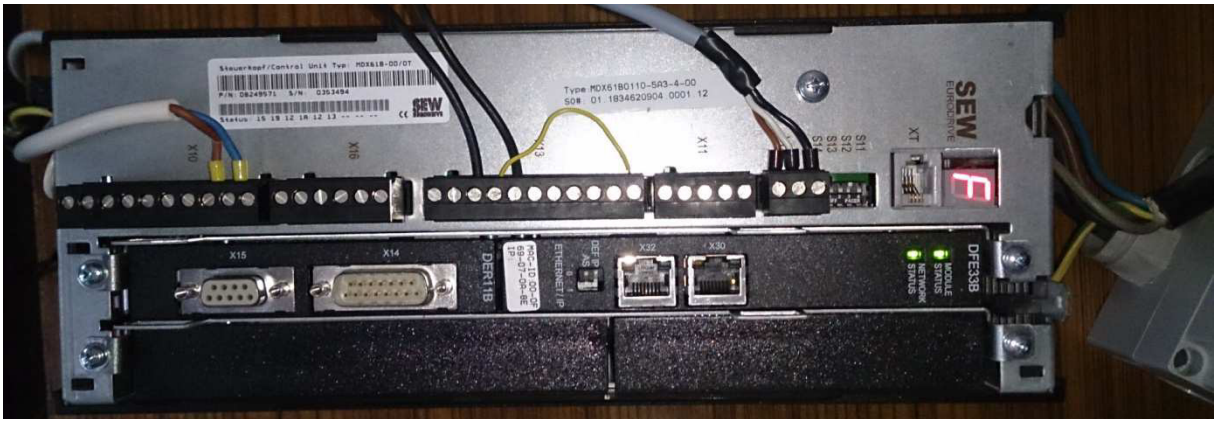

Abbildung 4: Anschlussmöglichkeiten des DFE33B

Es wurde eine Steuerung via TCP in einem lokalen Netzwerk gewählt, da sich diese nach der Recherche als einfachste Methode darstellt und nicht die Anschaffung weiterer Bauteile wie einer Modbuskarte erfordert. Auf die Möglichkeit einer Steuerung mittels Modbus soll allerdings kurz eingegangen werden, da SEW hierzu ein eigenes LabVIEW Programm anbietet.

4.16.3 Elektronikdaten

Option DFE33B		
 1455412875	Applikations-Protokolle	• **EtherNet/IP** (Ethernet Industrial Protocol) oder **Modbus/TCP** zur Steuerung und Parametrierung des Antriebsumrichters. • **HTTP** (Hypertext Transfer Protocol) zur Diagnose mittels Web-Browser. • **SMLP** (Simple MOVILINK Protocol), Protokoll, das von MOVITOOLS® MotionStudio genutzt wird. • **DHCP** (Dynamic Host Configuration Protocol) zur automatischen Vergabe der Adressparameter.
	Verwendete Port-Nummern	• 44818 EtherNet/IP (TCP) • 2222 EtherNet/IP (UDP) • 502 Modbus/TCP • 300 SMLP (TCP, UDP) • 80 HTTP • 67 / 68 DHCP
	Ethernet-Dienste	• ARP • ICMP (Ping)
	ISO / OSI-Schicht 1/2 **ISO / OSI-Schicht 4/5**	Ethernet II TCP/IP und UDP/IP
	Automatische Baudraten-erkennung	10 MBaud / 100 MBaud
	Anschlusstechnik	2 x RJ45 mit integriertem Switch und Auto-Crossing
	Adressierung	4 Byte IP-Adresse bzw. MAC-ID (00-0F-69-xx-xx-xx)
	Herstellerkennung (Vendor-ID)	• $013B_{hex}$ (EtherNet/IP) • "SEW-EURODRIVE" (Modbus/TCP)
	Hilfsmittel zur Inbetriebnahme	• Engineering-Software MOVITOOLS® MotionStudio ab Version 5.40 • Bediengerät DBG60B
	Firmware-Stand des MOVIDRIVE® MDX61B	Firmware-Stand 824 854 0.17 oder höher (→ Anzeige mit P076)

Abbildung 5: Auszug aus dem SEW Systemhandbuch [19]

Anhand des Systemhandbuchs von SEW (Abbildung 5) können die möglichen Protokolle eingesehen werden, die zur Datenübertragung genutzt werden können. In den nachfolgenden Kapiteln soll auf die einzelnen Protokolle und ihre Möglichkeiten eingegangen werden. Es werde Folgende Möglichkeiten von Protokollen erläutert:

- 44818 EtherNet/IP (TCP)
- 502 Modbus/TCP
- 300 SMLP (TCP, UDP)

Ergänzend zu der Programmierung soll eine Methode entworfen werden, mit der die Genauigkeit der eingestellten Winkel überprüft werden kann.

2. Grundlagen

2.1 Wireshark

Wireshark ist ein Netzwerkpaket Analysierer, auch Sniffer genannt. Wireshark greift die Netzwerkpakete eines Netzwerkes oder einer bestimmten Verbindung ab und versucht diese möglichst detailliert widerzugeben [8]. Mit neueren Versionen ist auch das Abgreifen von Daten in einem CANopen Netzwerk möglich.

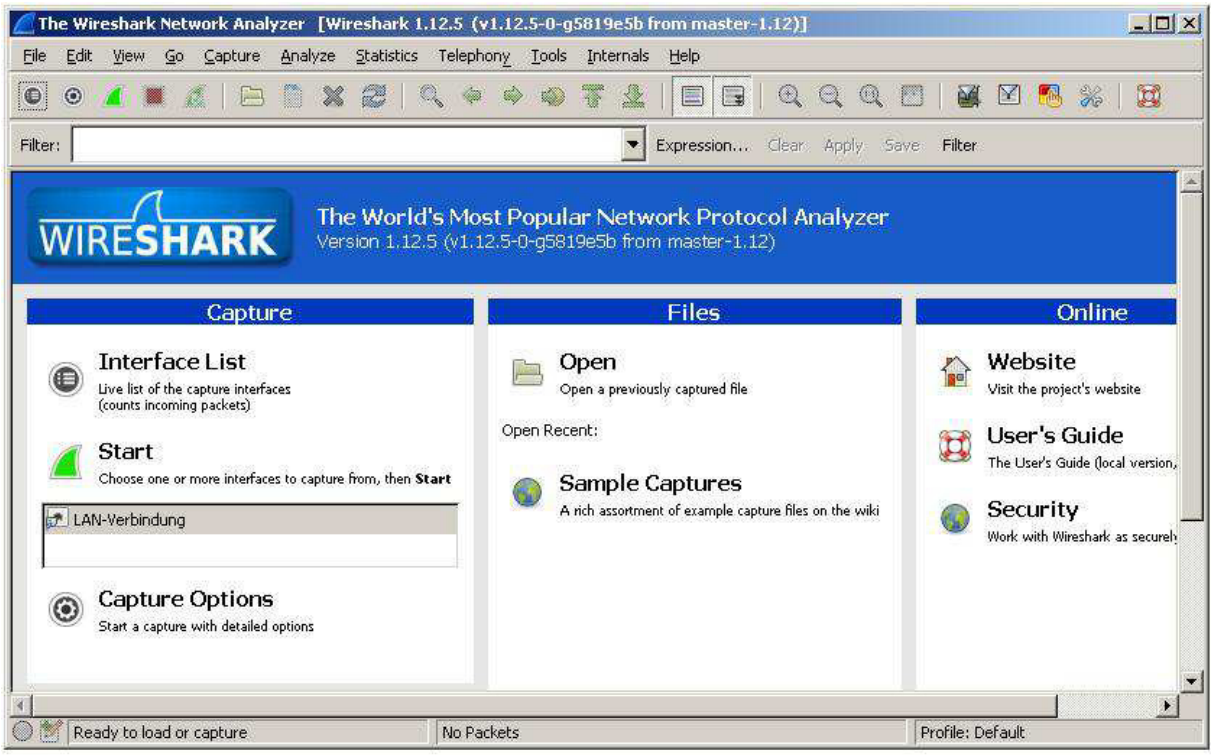

Abbildung 6: Wireshark Startbildschirm

Nachdem eine Verbindung gewählt wurde, die überwacht und mitgeschrieben werden soll, wird das Hauptprogramm gestartet. Hier werden nun die Datenpakete angezeigt und können im Einzelnen analysiert werden. Die Anzeige gliedert sich wie in der folgenden Tabelle.

No	→	Nummer des Pakets. Das erste aufgezeichnete Paket erhält die Nummer 1
Time	→	Ankunftszeit des Paketes (je nach Konfiguration Uhrzeit oder vergangene Zeit seit Beginn der Aufzeichnung)
Source	→	Absender des Paketes (je nach Konfiguration IP-Adresse, FQDN, Hostname oder MAC-Adresse)
Destination	→	Empfänger des Paketes (je nach Konfiguration IP-Adresse, FQDN, Hostname oder MAC-Adresse)
Protocol	→	Verwendetes Netzwerk-Protokoll, beispielsweise im Falle eines PingRequests typischerweise ICMP
Info	→	Kurze Beschreibung des Inhaltes des Paketes, beispielsweise im Falle eines Ping-Reply typischerweise „Echo (ping) reply“

Tabelle 1: Wireshark Gliederung [9]

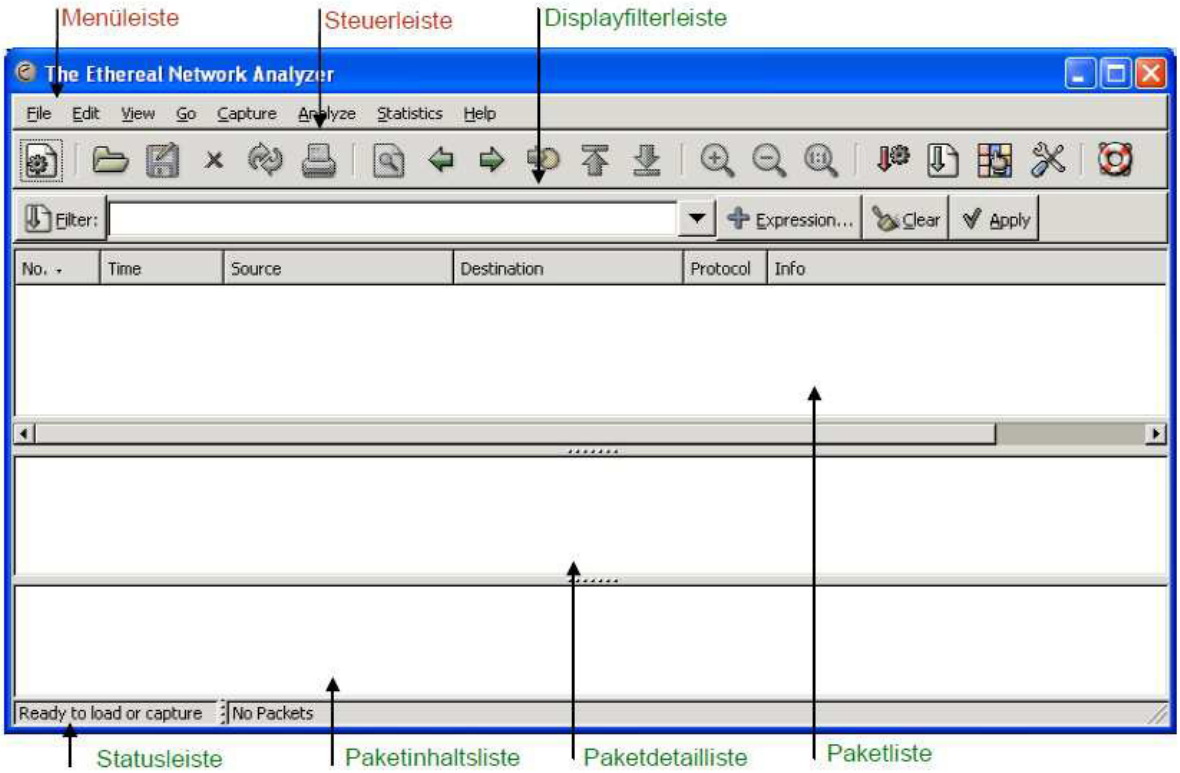

Abbildung 7: Wireshark Protokoll [18]

Im oberen Teil der Anzeige, der Paketliste, werden die abgegriffenen Netzwerkpakete angezeigt. Klickt man eines dieser Pakete an, so können im unteren Teil die Details des Pakets angezeigt werden. Die Detailanzeige unterscheidet sich zwischen der Paketdetailliste und der Paketinhaltsliste, in der die Daten als Hexadezimalcode angezeigt werden.

Wenn es nicht erwünscht ist, dass bestimmte Protokolle oder andere Daten angezeigt werden, kann der Mitschrieb über das Anbringen verschiedener Filter in der Displayfilterleiste bereinigt werden, sodass nur die gewünschten Pakete protokolliert werden. Die Filter unterliegen dabei in der Eingabe ähnlichen regeln wie in der Programmierung, so bedeutet ein „!“ eine Negation des nachfolgenden Wertes der Art „nicht“, während „==“ eine Eindeutigkeit der Art „ist genau gleich“ beschreibt.

Filter Eingabe	Beschreibung
ip.addr == 192.168.10.24	Zeigt jeden Traffic von oder zu der IP Adresse 192.168.10.24 an
! (ip.addr == 192.168.10.24)	Jeder Traffic, der die IP Adresse 192.168.10.24 beinhaltet wird ausgeblendet

Tabelle 2: Beispiele für Wireshark Filter

2.2 Netzwerktheorie

2.2.1 Transmission Control Protocol TCP

Das Transmission Control Protocol (TCP) gilt zu den transportorientierten Netzwerkprotokollen und dient der Definition für den Datenaustausch zwischen den einzelnen Stationen in einem Netzwerk. Es stellt eine End-to-End Verbindung zwischen den Sockets, also den Endpunkten, im Netzwerk her. Diese Endpunkte sind zumeist durch ihre Internet Protokoll (IP) Adresse definiert, da das TCP auf der Internet Schicht im Open System Interconnection Modell (OSI) aufbaut, daher wird auch vom TCP/IP gesprochen.

Schicht	Dienst / Protokolle / Anwendung			
Anwendung	HTTP	IMAP	DNS	SNMP
Transport	TCP		UDP	
Internet	IP (IPv4 / IPv6)			
Netzzugang	Ethernet, …			

Abbildung 8: OSI Schichtmodell (Ausschnitt) [10]

TCP ist ein Vollduplex Datenübertragungssystem, was bedeutet, dass Daten in beide Richtungen gleichzeitig ausgetauscht werden können. Zudem kann das TCP bei Datenverlust diesen automatisch erkennen und beheben.

Zum Verbindungsaufbau wird der sogenannte drei Wege Handschlag (engl. 3-way-handshake) (Abbildung 9) verwendet. In diesem sendet der Anfragende Part, auch Client genannt, eine TCP-Verbindungsanforderung, die sogenannte synchronize massage (SYN). Der angefragte TCP-Server antwortet darauf mit einer Bestätigung, bestehend aus einer weiteren Synchronisation und einer Bestätigungsnachricht, diese wird acknowledgement (ACK) genannt, worauf der Client eine weitere ACK Nachricht zur widerholten Bestätigung der Verbindung sendet. Ist eine Verbindung aufgebaut, können Daten zwischen beiden Teilen der Verbindung gleichberechtigt ausgetauscht werden. In einem TCP Netzwerk gibt es nach dem Kommunikationsaufbau keinen Unterschied zwischen Server und Client.

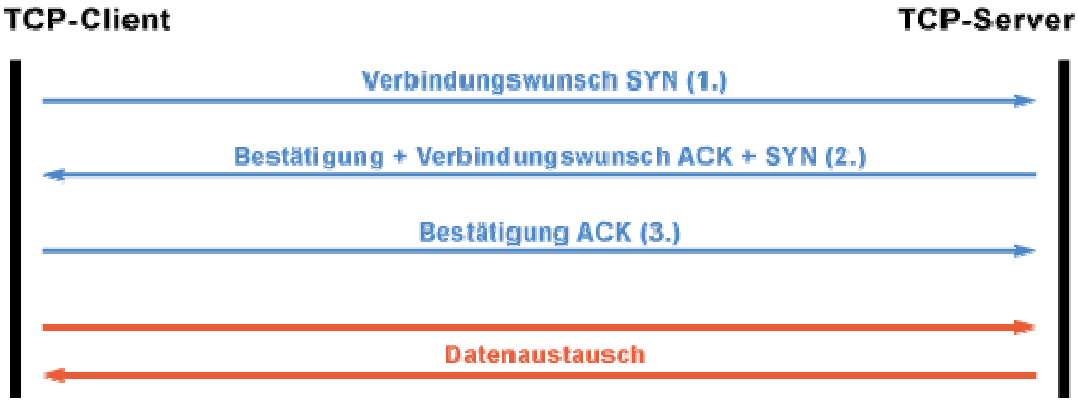

Abbildung 9: Aufbau der TCP Verbindung [10]

Daten werden in einer TCP-Verbindung mittels sogenannter Ports bestimmten Anwendungen zugeordnet. Diese Portnummern sind klar genormt durch die Internet Assigned Numbers Authority (IANA) und die Internet Corporation for Assigned Names and Numbers (ICANN).

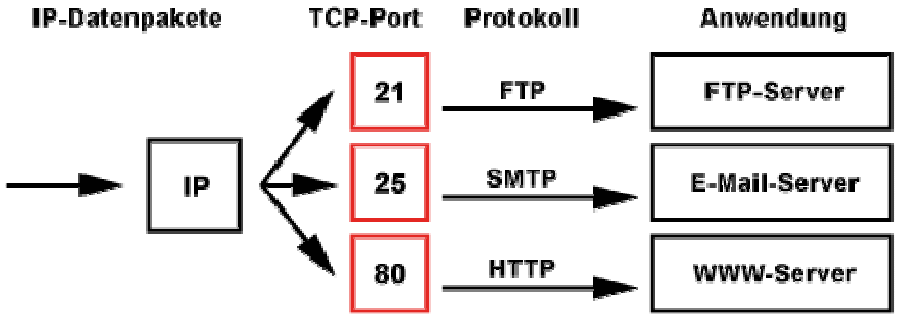

Abbildung 10: Port zuordnung für TCP/IP [10]

Man unterscheidet zwischen den „Well Known Ports“, den „Registered Ports“ und den „Dynamically Allocated Ports“.

Die „Well Known Ports“ sind standardisierte und fest zugeordnete Ports, die eine einfache Kommunikation über TCP ermöglichen. Sie sind unabhängig vom Betriebssystem und Zielsystem anwendbar. So ist wie oben zu sehen die Portnummer 25 dem Simple Mail Transfer Protocol (SMTP) zuzuschreiben. Dieser Dienst ermöglicht eine einfache Kommunikation zwischen Mailservern und Clients um die E-Mailkommunikation zu vereinfachen.

Als „Registered Ports“ werden die Portnummern 1024 bis 49151 bezeichnet. Diese sind nicht alle belegt und können bei der IANA/ICANN für Dienste reserviert werden. So ist beispielsweise der Port 23399 für Anwendungen von Skype reserviert. Die Dienste in diesem Portnummernbereich sind nicht auf allen Systemen standardisiert.

„Dynamically Allocated Ports“ sind dynamisch vergebene Dienste und können jederzeit von Clients benutzt werden um eigene Dienste anzusprechen.

TCP Datenpakete sind immer gleich aufgebaut, sie bestehen aus dem Datenbereich und dem Headerbereich. Im TCP-Header stehen die verbindungsrelevanten Daten, die zum Aufbau der Verbindung im Netzwerk nötig sind.

<table>
<tr><td colspan="2">Absender-Port</td><td colspan="2">Empfänger-Port</td></tr>
<tr><td colspan="4">Sequenznummer</td></tr>
<tr><td colspan="4">Bestätigungsnummer</td></tr>
<tr><td>Daten-Offset</td><td>Reserviert</td><td>Code</td><td>Fenster</td></tr>
<tr><td colspan="2">Prüfsumme</td><td colspan="2">Dringlichkeitszeiger</td></tr>
<tr><td colspan="2">Optionen</td><td colspan="2">Füllzeichen</td></tr>
<tr><td colspan="4">Daten</td></tr>
</table>

Abbildung 11: Aufbau des TCP Headers [11]

Feldinhalt	Bit	Beschreibung
Quell-Port (Source-Port)	16	Hier steht der Quell-Port, von der die Anwendung das TCP-Paket verschickt. Bei einer Stellenanzahl von 16 Bit beträgt der höchste Port 65.535.
Ziel-Port (Destination-Port	16	Hier steht der Ziel-Port, über welchen das TCP-Paket der Anwendung zugestellt wird. Bei einer Stellenanzahl von 16 Bit beträgt der höchste Port 65.535.
Sequenz-Numer	32	Bei jeder TCP-Verbindung werden Nummern zwischen den Kommunikationspartner ausgehandelt. Während der Verbindung werden diese Nummern verwendet um die TCP-Pakete eindeutig zu identifizieren.
Acknowledgement-Nummer	32	Alle Datenpakete werden bestätigt. Dazu dient das ACK-Flag und die Acknowledgement-Nummer, die sich aus der Sequenz-Nummer und der Anzahl von empfangenen Bytes errechnet. Damit kann der Sender feststellen, ob die Daten beim Empfänger vollständig angekommen sind.
Data Offset	4	Hier steht die Anzahl der 32-Bit-Blöcke des TCP-Headers. Die Mindestmenge beträgt 5.
Reserviert	6	Dieser Bereich ist auf 000000 gesetzt und für Erweiterungen des TCP-Headers gedacht.
Flags	6	Kennzeichnung bestimmter für die Kommunikation und Weiterverarbeitung der Daten wichtiger Zustände (URG, ACK, PSH, RST, SYN, FIN).
Window-Größe	16	Der Empfänger sendet dem Sender in diesem Feld die Anzahl an Daten, die der Sender senden darf. Dadurch wird das Überlaufen des Empfangspuffers beim Empfänger verhindert. Den Vorgang nennt man Windowing und dient der Datenflusssteuerung.
Check-Summe	16	Dieses Feld dient der Kontrolle von Header- und Datenbereich.
Urgent-Pointer	16	Zusammen mit der Sequenz-Nummer gibt dieser Wert die genaue Position der Daten im Datenstrom an. Der Wert ist nur gültig, wenn das URG-Flag gesetzt ist.
Optionen/Füllbits (Options/Padding), jeweils 32 Bit lang		Dieses Feld beinhaltet optionale Informationen. Um die 32-Bit-Grenze einzuhalten wird das Options-Feld mit Nullen aufgefüllt.

Tabelle 3: Bedeutung der Felder im TCP Header [10]

Der in Abbildung 11 dargestellte Aufbau des TCP Headers kann anhand eines mit Wireshark erstellten Protokolls von einer Verbindung vom Rechner mit dem Frequenzumrichter nachvollzogen werden. Hier wurde mittels Movitools eine Verbindung aufgebaut.

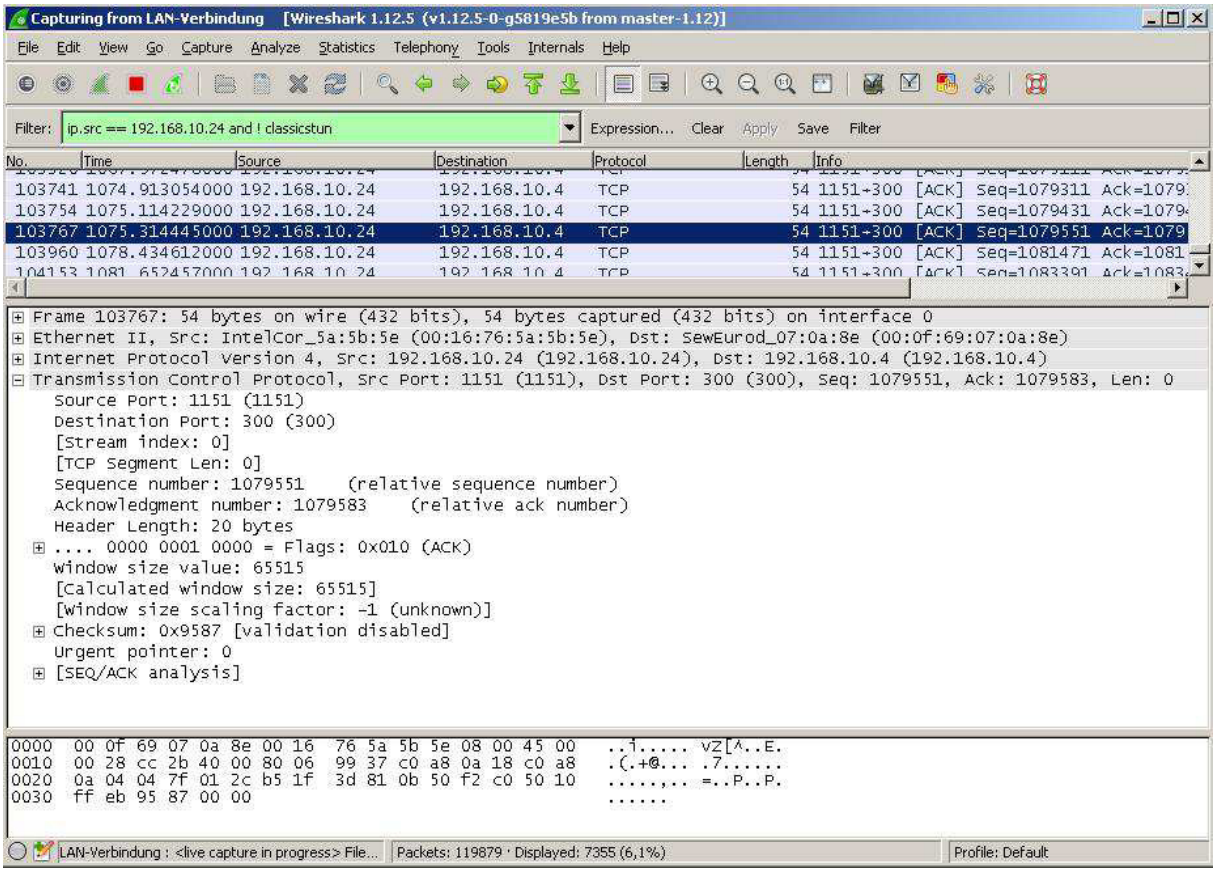

Abbildung 12: Aufbau des TCP-Headers in Wireshark

Es ist zu sehen, dass der Source Port (Quellport) 1151 ist, somit handelt es sich um ein TCP/UDP Protokoll, welches von der IP-Adresse 192.168.10.24 an die Zieladresse 192.168.10.4, also an den Frequenzumrichter auf den Destination Port (Zielport) 300 versendet wird. Dieser Port ist wie weiter unten beschrieben ein SEW eigenes Protokoll, das Simple Movilink Protocol (SMLP).

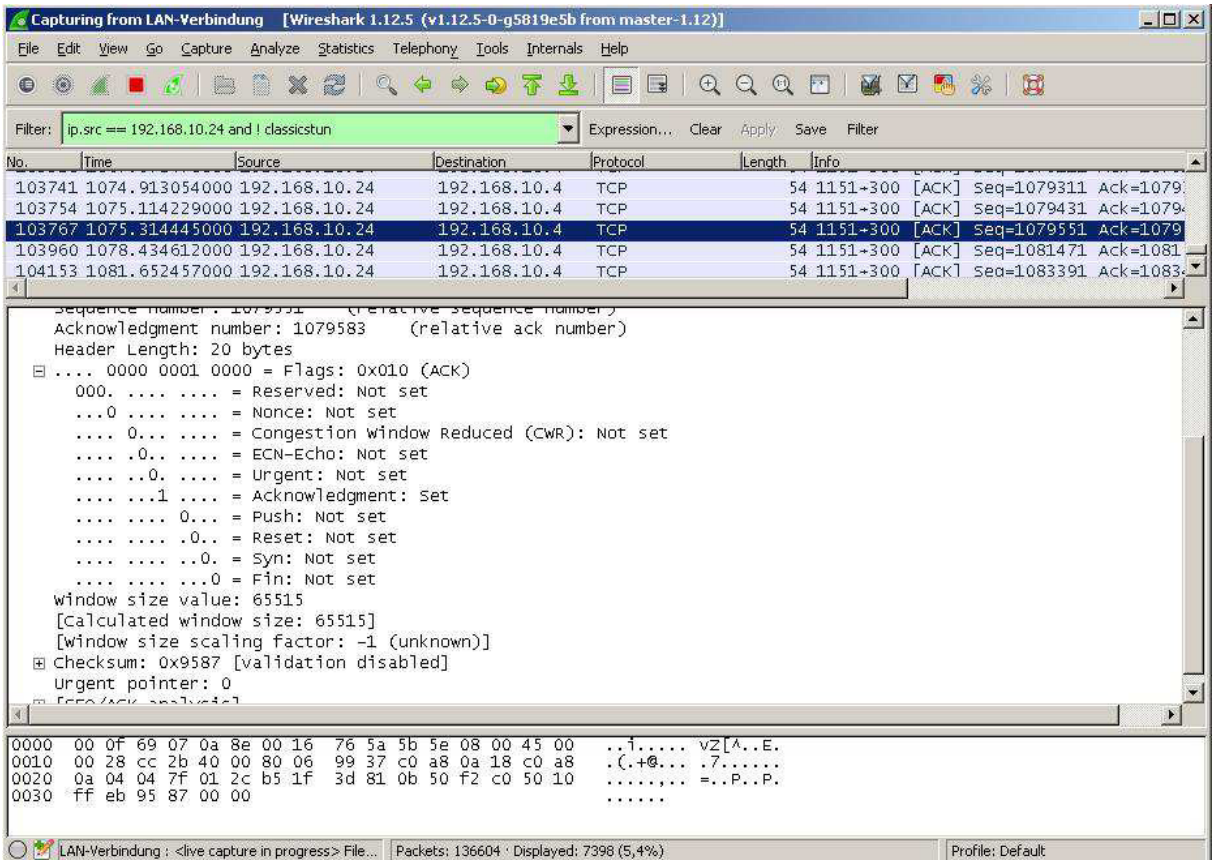

Abbildung 13: TCP Flags im Wireshark Protokoll

Als einzige Flag ist das Acknowledgment gesetzt (Abbildung 13), es handelt sich daher bei der abgebildeten Nachricht um eine Verbindungsanfrage.

Zusätzlich zu den Headerdaten werden zudem die Daten der IPv4 Verbindung sowie die Hardwaredaten der Ethernet II Verbindung angezeigt. Hieraus lassen sich die Hardwareadresse der Netzwerkpartner erkennen, beispielsweise die Sourceadresse *IntelCor_5a:5b:5e* die auf den IntelCore Netzwerkchip des Rechners verweist.

2.2.2 Simple Movilink Protocol SMLP

Das Simple Movilink Protocol ist ein SEW eigenes TCP Netzwerk Kommunikationsprotokoll, welches auf der Port Nummer 300 liegt und somit nicht zu den standardisierten Ports gehört. Mit dem SMLP können TCP-Daten von Movitools an den Frequenzumrichter DFE33B übermittelt werden.

2.2.3 EtherNet Industrial Protocol EtherNet/IP

Ethernet Industrial Protocol (EtherNet/IP) ist ein offener Standard für industrielle Netzwerke, der zyklische Eingang/Ausgabe-Nachrichtenübertragung (E/A) sowie azyklische (explizite) Nachrichtenübertragung unterstützt [12]. Das Protokoll hat die Portnummer 44818. Das EtherNet/IP setzte auf dem TCP und User Datagram Protocol (UDP) auf. Es wird Common Industrial Protocol (CIP) genannt und befindet sich in den Darstellungs-, Sitzungs-, Anwendungs-, sowie Geräteprofil-Ebenen der OSI Schichten.

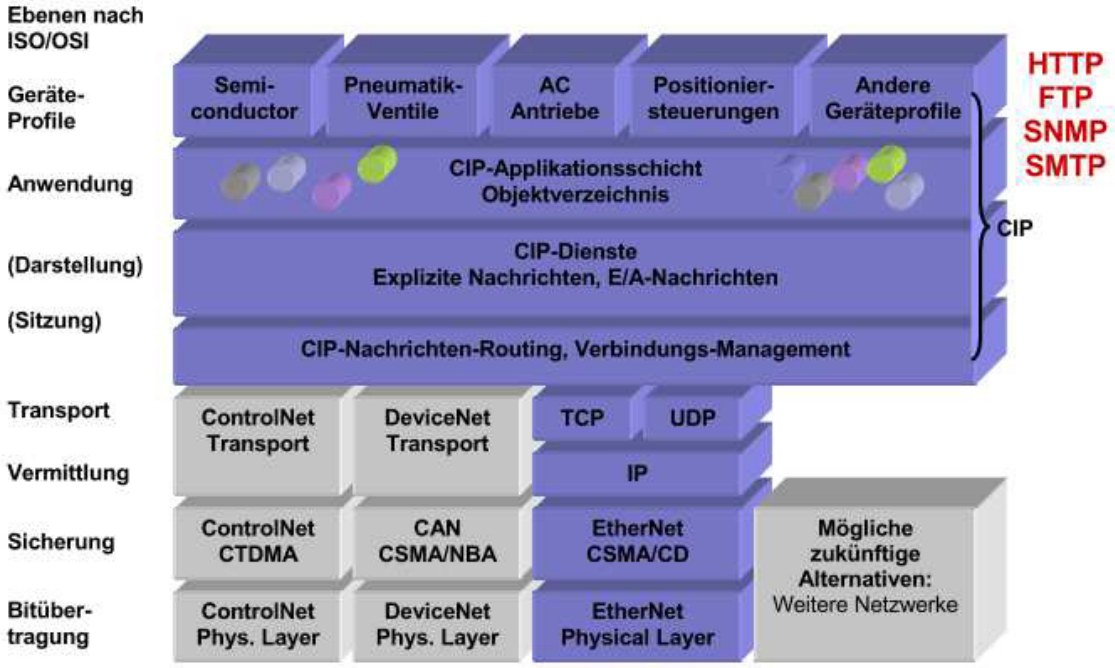

Abbildung 14: OSI Schichtmodell mit EtherNet/IP CIP-Schicht [12]

Man unterscheidet beim CIP zwei Arten von EtherNet/IP Übertagungen, zum einen explizite Informationsübertragungen und zum anderen Echtzeit-E/A-Datenübertragungen (implizit).

Explizite Datenübertragung befasst sich mit der Übertragung nicht zeitkritischer Informationen von einem Sender zu einem einzelnen Ziel. Die Daten werden mittels des TCP/IP-Protokolls übertragen, da sie zumeist größere Datenpakete umfassen. Sie dienen einfachen Lese- oder Schreibbefehlen.

Implizite E/A-Daten werden zum Echtzeittransfer verwendet und können von einem Sender an eine beliebige Anzahl Empfänger gesendet werden. Sie dienen zum Steuern von E/A-Geräten wie Sensoren oder Frequenzumrichtern. Aufgrund der kleinen Datengröße und des erhöhten benötigten Datendurchsatzes wird das UDP/IP-Protokoll verwendet.

EtherNet/IP Übertragungsarten	Nachrichtenart	Beschreibung	Beispiel
Information	Explizit	Nicht zeitkritiche Informationsdaten	Lesen/Schreiben von Daten mittels Nachrichtenbefehlen
E/A-Daten	Implizit	Echtzeit-E/A-Daten	Steuern von Echtzeitdaten über ein dezentrales E/A-Gerät, Austausch von Echtzeitdaten zwischen Steuerungen

Tabelle 4: Übertragungsarten für EtherNet/IP [12]

Man unterscheidet bei E/A-Geräten zwischen Scannern und Adaptern. Diese sind am ehesten mit einer Master-Slave-Verknüpfung vergleichbar. Adapter sind beispielsweise Sensoren, Aktoren und Frequenzumrichter. Diese Geräte müssen zunächst von einem Scanner aktiv angesprochen werden, es können hierbei mehrere Geräte parallel kontrolliert und angesprochen werden. In der hier beschriebenen Anordnung würde dem Rechner die Scanner- und dem Frequenzumrichter DFE33B die Adapter Rolle zu fallen.

2.2.4 Nagle-Algorithmus

Der Nagle-Algorithmus dient in Netzwerken dazu, zu kleine Datenpakete im TCP Teil zu verhindern. Dies ist in normalen Netzwerken wichtig, da durch zu kleine Pakete der Datenduchsatz verringert wird, somit leidet die Übertragungsgeschwindigkeit. Für die Verminderung des Duchsatzes ist der Aufbau von TCP Nachrichten verantwortlich, da diese mit einem festen Header erstellt werden und bei zu kleinen Paketen die Relevanz des Header die Relevanz der Nachricht überwiegen kann.

Der Nagle-Algorithmus basiert auf zwei Grundprinzipien.

- Wenn ein Paket voll ist, soll dieses gesendet werden
- Wenn ein Paket nicht voll ist, soll solange gewartet werden, bis alle bestätigten Pakete abgearbeitet sind

Die normale Wartezeit des Nagle-Algorithmus beträgt 200 Millisekunden [ms]. Dies führt im Rahmen der Programmierung mit LabVIEW zu Problemen, da wie später erläutert wird, die Daten mittels TCP alle 80 ms übermittelt werden sollen. Um dies zu realisieren muss der Algorithmus umgangen werden. Hierzu gibt es zum Einen die Möglichkeit direkt in die windowsinternen Netzwerkprotokolle einzugreifen, was mit erheblichem Aufwand und Kenntnissen über Netzwerk und Systemprogrammierung verbunden wäre. Zum Anderen bietet ein LabVIEW VI, welches von National Instrument zum Download angeboten wird, die Möglichkeit den Nagle-Algorithmus für einzelne TCP Verbindungen zu umgehen.

2.3 Modbus

Das Modbus-Protokoll wurde von der Firma Modicon für den Datenverkehr mit ihren Controllern entwickelt [13]. Es basiert auf einem klassischen Master/Slave-Protokoll. In diesem steuert ein Single-Master, zum Beispiel ein Computer, einen einzelnen oder eine Reihe von Slave Geräten mittels Telegrammen. Im Gegensatz zum TCP sind hier Master und Slave nicht gleichberechtigt. Slave Geräte sind nicht berechtigt ohne Anfrage des Masters Telegramme zu senden.

Abbildung 15: Modbuskarte CIFX 50-RE von Hilscher [14]

Man unterscheidet zwischen drei Modbusprotokollen

- Modbus/ASCII
- Modbus/RTU
- Modbus/TCP

Jeder Teilnehmer im Netzwerk muss eine eindeutige Adresse besitzen, damit er vom Master identifiziert und direkt angesprochen werden kann.

2.3.1 Modbus/ASCII

Im ASCII Modus werden die Daten mittels ASCII-Code übertragen und nicht in Binärfolge. Das bedeutet, die Befehle sind in Klartext verfasst und können direkt gelesen werden. Die Übertragung von ASCII-Code erfordert allerdings eine höhere Rechenleistung, wodurch der Datendurchsatz im Vergleich zum RTU oder TCP Modus deutlich geringer ist.

Das ASCII Telegramm setzt sich dabei wie in Tabelle 5 dargestellt zusammen.

Start	Adresse	Funktion	Daten	LR-Check	Ende
1 Zeichen	2 Zeichen	2 Zeichen	n Zeichen	2 Zeichen	2 Zeichen (CRLF)

Tabelle 5: Aufbau des Modbus/ASCII Telegramm

Als Startzeichen steht normalerweise ein „:“, die Adresse bilden zwei ASCII-Zeichen, die das Zielgerät identifizieren. Im Datenfeld werden die auszuführenden Befehle in Klartext eingegeben. Diese Befehle sind vom Zielgerät abhängig. Im Check Feld wird eine Längenparitätsprüfung (longitudinal redundancy check, LRC) durchgeführt. Bei dieser Prüfung wird über eine bestimmte Anzahl von gesendeten Daten eine Prüfsumme gebildet und ein Prüfdaten-

satz gesendet. Werden diese nun miteinander verglichen, so sollen sie gleich groß sein. Hierdurch können 1 Bit Datenfehler erkannt werden.

2.3.2 Modbus/RTU

Beim Modbus/RTU (Remote Terminal Unit) werden die Daten in Binärfolge übertragen, dies ermöglicht einen hohen Datendurchsatz. Jedoch ist es hierbei nötig eine dauerhafte Verbindung zu gewährleisten, da bei einem Abbruch der Verbindung Daten nicht vervollständigt werden können und der Slave im Netzwerk eine Fehlermeldung auslöst. Den Start des RTU-Telegrammes bildet eine Sendepause von mindestens drei Zeichen. Die Wartezeit selber ist somit von der Übertragungsrate abhängig, in der die drei Wartezeichen gesendet werden können. Die Adresse bildet ein 1 Byte, also 8 Bit Datensatz, der ähnlich wie im ASCII Modus den Empfänger identifiziert. Im Funktionsfeld wird dem Slave übermittelt, welche Art von Befehl ausgeführt werden soll. Das Datenfeld enthält die auszulesenden Daten. Der Slave füllt dieses Datenfeld aus und sendet es an den Master zurück. Im Gegensatz zum LRC wird hier eine zyklische Redundanzprüfung (cyclic redundancy check, CRC) durchgeführt. Bei diesem Verfahren wird für jedes Datenwort ein CRC-Wert berechnet und nach dem Senden vom Empfänger überprüft. Stimmen die Werte überein, so ist davon auszugehen, dass kein Sendefehler aufgetreten ist.

Start	Adresse	Funktion	Daten	CR-Check	Ende
Wartezeit (min. 3,5 Zeichen)	1 Byte	1 Byte	n Byte	2 Byte	Wartezeit (min. 3,5 Zeichen)

Tabelle 6: Aufbau des Modbus/RTU Telegramm

2.3.3 Modbus/TCP

Im Modbus/TCP werden TCP/IP-Pakete versendet, um die Daten zu übermitteln. Die Kommunikation findet somit gemäß der Standards einer TCP Verbindung statt. Die Kontrolle wird hierbei wie in einem regulären TCP-Paket durch die Verbindung selbst gewährleistet und es sind keine Kontroll-Bits, wie in den anderen beiden Modi nötig. Modbus/TCP ist in der IEC 61158 genormt und liegt auf dem TCP-Port 502.

Transaktionsnummer	Protokollkennzeichen	Zahl der noch folgenden Bytes	Adresse	Funktion	Daten
2 Byte	2 Byte (immer 0x0000)	2 Byte (n+2)	1 Byte	1 Byte	n Byte

Tabelle 7: Aufbau des Modbus/TCP Telegramm

2.4 LabVIEW

Die oberflächlichen Grundlagen von LabVIEW wurden bereits in den Arbeiten CAN-Bus-Ansteuerung für einen elektrischen Wellenkanalantrieb (*Hausarbeit HAW Hamburg, 2014*), sowie Inbetriebnahme und Erprobung einer Steuerung für einen Wellenkanalantrieb (*Hausarbeit HAW Hamburg, 2014*) behandelt. Daher wird hier nur kurz auf diese Grundlagen eingegangen und die TCP Netzwerk Programmierung mit LabVIEW insbesondere betrachtet.

Bei LabVIEW werden Programmbausteine, sogenannte virtuelle Instrumente (Virtual Instrument, VIs), auf einer Arbeitsfläche platziert und die Ein- und Ausgänge mittels Drähten verbunden. Eingaben finden über das Frontpanel statt, während die eigentlichen Bausteine auf dem Blockdiagramm zusammen gefügt werden.

Zur Programmierung einer Netzwerkkommunikation mit Schreibbefehl in LabVIEW werden drei VIs benötigt, diese sollen im Folgenden erläutert werden.

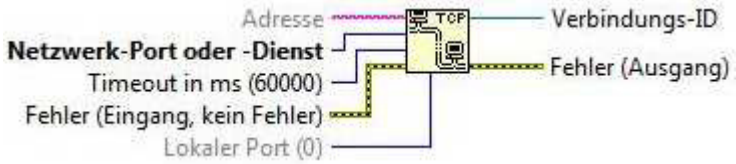

Abbildung 16: TCP Verbindung herstellen [16]

Das in Abbildung 16 dargestellte VI dient zum Aufbau einer TCP-Verbindung zwischen dem Computer und einem Zielgerät. Am Adresseingang wird dem Funktionsbaustein die Zieladresse mitgeteilt. Diese kann als IP-Adresse oder als Computername eingegeben werden. Den Netzwerkport bildet die Portnummer, über die eine Verbindung aufgebaut werden soll. Diese ist wie in Kapitel 3.2 unter dem Abschnitt TCP aufgeführt definiert. Der Timeout legt fest wie lange auf eine Antwort gewartet werden soll, bevor ein Fehler ausgegeben wird. Wird dieser Wert auf -1 gesetzt so wird kein Fehler ausgegeben. Die Voreinstellung liegt bei 60000 ms (1 min). Der Timeout sollte nicht als Eingabe sondern als Konstante festgelegt werden. Der Fehlereingang wird wie in allen VIs Mit einem Fehlerdisplay belegt und die sogenannte Errorline bis zum Ende des Programms durchgezogen. Der lokale Port dient zum Identifiezieren des Ausgangsdienstes, bei einer Voreinstellung von 0 wird die Portnummer automatsch festgelegt. Die Ausgangsklemme gibt die Verbindungs-ID aus, die an die anderen TCP-VIs weitergegeben wird, um diese mit den Verbindungsdaten zu versorgen.

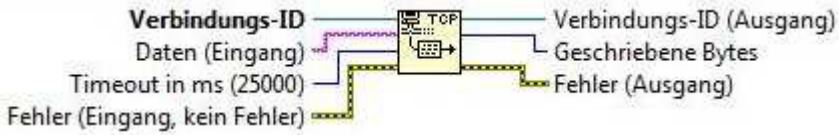

Abbildung 17: TCP Schreiben [16]

Das „TCP Schreiben“-VI (Abbildung 17) benötigt als Eingang die Verbindungsdaten, die in Form der Verbindungs-ID vom oben beschriebenen „Verbindung herstellen“-VI bereitgestellt werden. Des Weiteren werden die Daten als Text eingegeben und durch das VI in ein TCP-Datenpaket umgewandelt und gesendet. Der Ausgang „Geschriebene Bytes“ gibt die Länge des Datensatzes wider, hierdurch kann überprüft werden, ob das TCP-Paket korrekt übermittelt wurde. Die Eingänge für den Timeout und den Fehler sind belegt, wie beim Herstellen der TCP Verbindung.

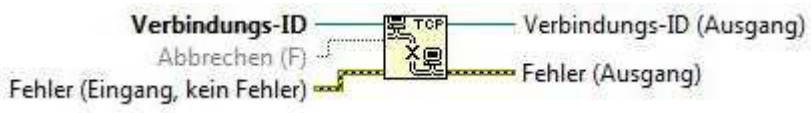

Abbildung 18: TCP Verbindung trennen [16]

Zum Trennen der TCP-Verbindung wird das VI aus Abbildung 18 verwendet. Dieses wird ebenfalls mit der Verbindungs-ID und zudem mit einem booleschen Element belegt, welches den Befehl zum Trennen der Verbindung ein- oder ausschaltet.

Neben diesen drei VI gibt es noch weitere, die zum Lesen von TCP Nachtrichten und zum Abrufen von Internet-Daten dienen. Da diese für die spätere Programmierung allerdings nicht relevant sind, werden sie hier nicht weiter betrachtet. Die Hilfe von LabVIEW gibt hierfür Hinweise, wie diese VIs verwendet werden können.

3. Programmierung

3.1 Vorgehen zur Programmierung

Das Ziel der Programmierung ist es, die von Movitools über TCP/IP geschickten Befehle nachzubilden und so eine Steuerung des Motors mittels LabVIEW zu ermöglichen. Zu diesem Zweck wird die Befehlsstruktur mit Wireshark analysiert und die einzelnen Befehle ausgelesen und nachgebildet.

Im ersten Schritt wird mittels Movitools eine Verbindung zum Frequenzumrichter aufgebaut. Hierzu wird der Rechner mittels eines LAN-Kabels mit dem Frequenzumrichter DFE33B verbunden. Anschließend wird über den Button „Scan“ und die Einstellung, dass nach einer Ethernet-, also einer Netzwerkverbindung gesucht werden soll, eine Kommunikation zwischen den beiden Geräten aufgebaut.

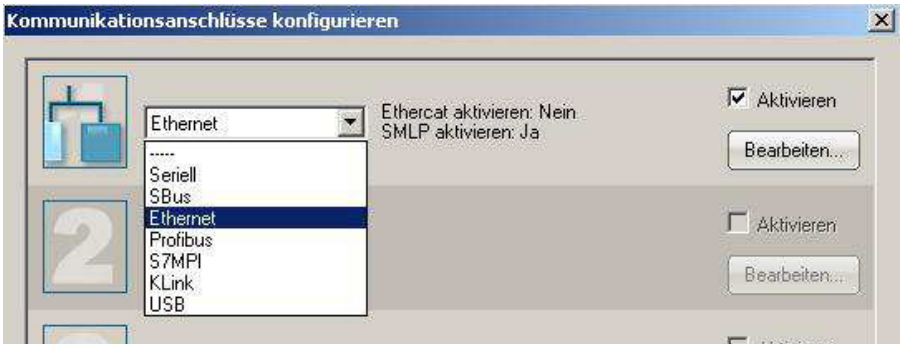

Abbildung 19: Movitools, Auswahl der Scan Verbindung

Hierbei ist zu beachten, dass sowohl der Computer als auch der Frequenzumrichter in dem gleichen IPv4 Bereich liegen müssen. Dies bedeutet dass beide Netzwerkparteien eine IP-Adresse mit dem gleichen Aufbau haben müssen. Zum Beispiel wie hier in der Form *192.168.10.XXX*, 192.168.10.24 für den Computer und 192.168.10.4 für den DFE33B.

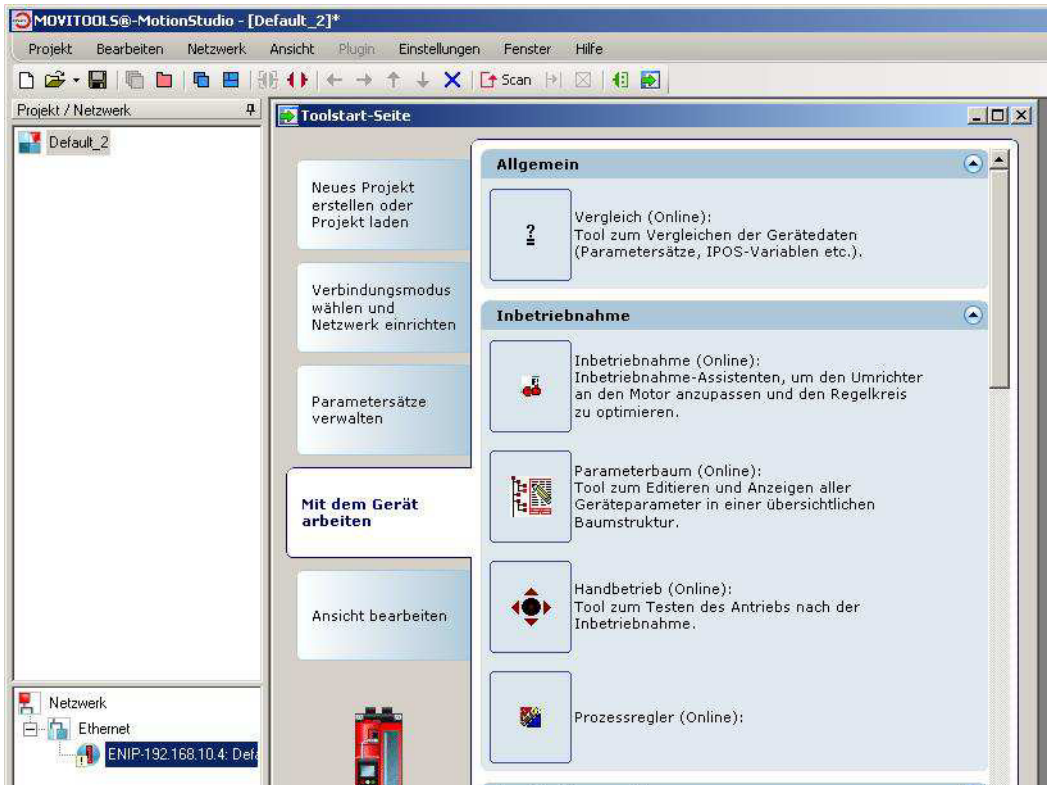

Abbildung 20: Movitools Aufbau der TCP Ethernetverbindung

Ist die Verbindung aufgebaut, ist dieses bereits im Wireshark Protokoll zu erkennen. Wie in Abbildung 21 zu sehen ist, senden beide Parteien des Netzwerkes abwechselnd eine „Binding Requests“, die zu einem „Classic-Stun“, welches ein TCP-Unterprotokoll ist, gehört. Dieses sind einfache Anfragen, ob die Verbindung weiterhin besteht.

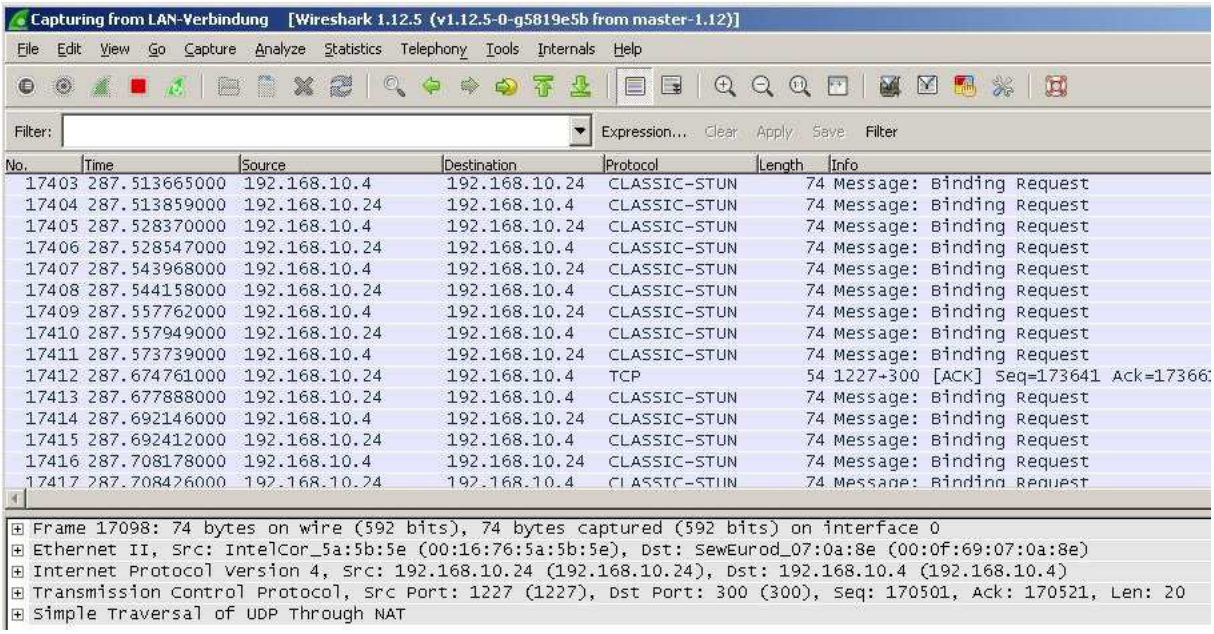

Abbildung 21: Wireshark Binding Request

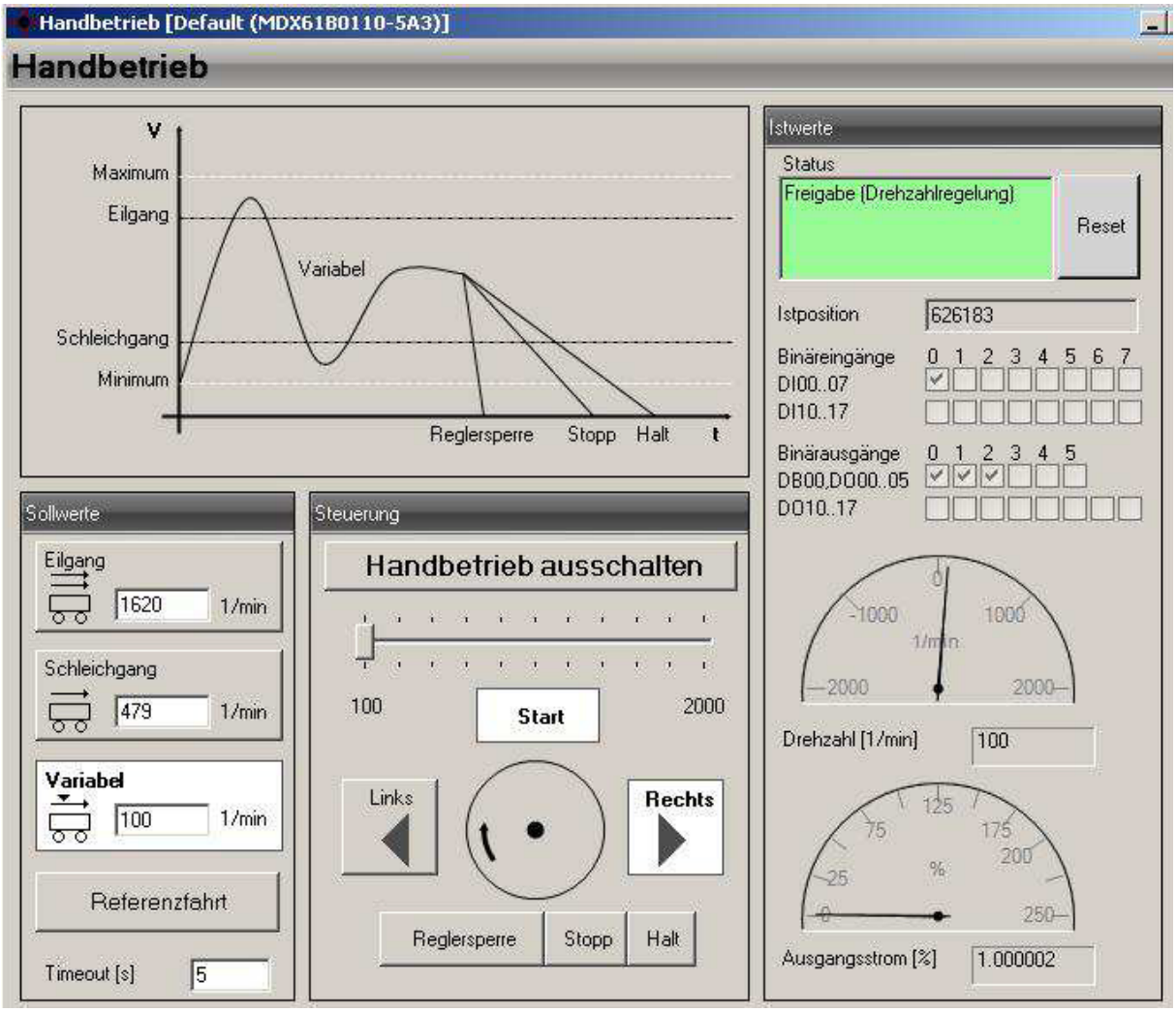

Abbildung 22: Handbetrieb von Movitools

Anschließend wird das Menü für den Handbetrieb aufgerufen und hier eine variable Drehzahl von 100 min^{-1} eingestellt (Abbildung 21). Über den Button „Handbetrieb einschalten“ wird nun der eigentliche Handbetrieb gestartet. Hierdurch wird eine TCP Nachricht an den Frequenzumrichter geschickt. Diese wird nun mittels Wireshark abgegriffen und im Protokoll festgehalten. Da die Classic-Stun Nachrichten allerdings einen Großteil des Protokolls einnehmen und in einem sehr kurzen Zeitintervall gesendet werden, müssen diese ausgefiltert werden, um die Steuerbefehle von Movitools auslesen zu können. Zudem interessieren zu diesem Zeitpunkt nur die ausgehenden Nachrichten an den DFE33B. Hierzu wird der Filter wie folgt eingegeben.

$$ip.src == 192.168.10.24\ and\ !\ classicstun$$

Es werden nun nur noch Nachrichten, die von der IP-Adresse 192.168.10.24, also vom Computer aus gesendet werden (ip.src) und keine Classic-Stun Nachrichten mehr angezeigt. Nachdem der Filter angebracht ist lässt sich die Nachricht zum einschalten des Handbetriebs ermitteln und auslesen (Abbildung 23).

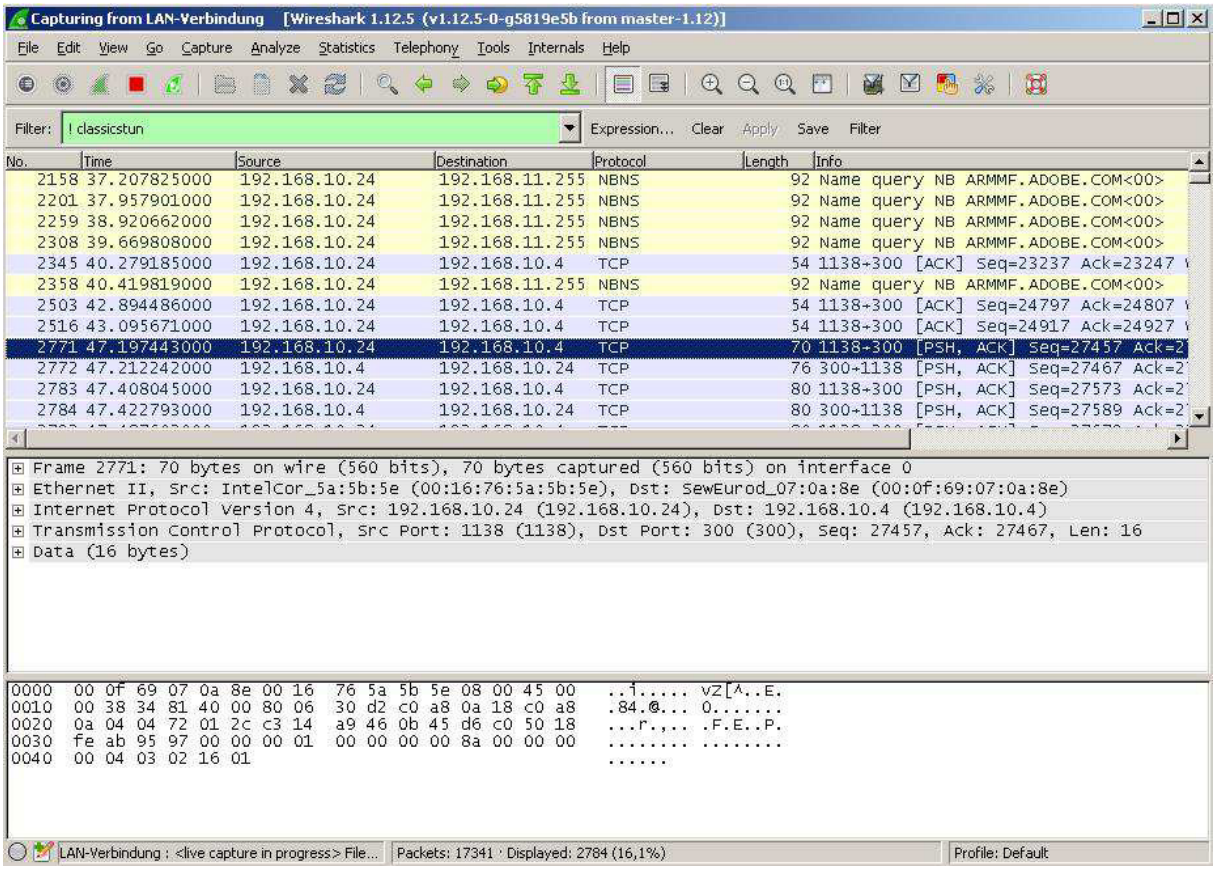

Abbildung 23: Wireshark Protokollierung des Handbetrieb

Über den „Start“ Button wird der Motor nun mit der eingestellten Drehzahl von 100 min^{-1} gestartet. Im Netzwerkprotokoll lassen sich die einzelnen Pakete auslesen, die dem Frequenzumrichter die Daten übermitteln (Abbildung 24). Anhand des Mitschriebes erkennt man, dass die Übermittlung der Drehzahl aus dem vorherigen Startbefehl des Handbetriebs und einem angehängten 2 Byte langen Bereich besteht. Der angehängte Bereich entspricht hierbei der eingestellten Drehzahl in Hexadezimaler Schreibweise. Es ist allerdings zu beachten, dass der Motor pro Hexdiget 0,2 min^{-1} schneller dreht. Somit berechnet sich die Hexadezimale Zahl am Ende des Datenpakets wie folgt:

$$100\ min^{-1} \div 0{,}2 = 500 = hex\ 1F4$$

Diese Datenpakete werden alle 0,07 bis 0,1 Sekunden gesendet.

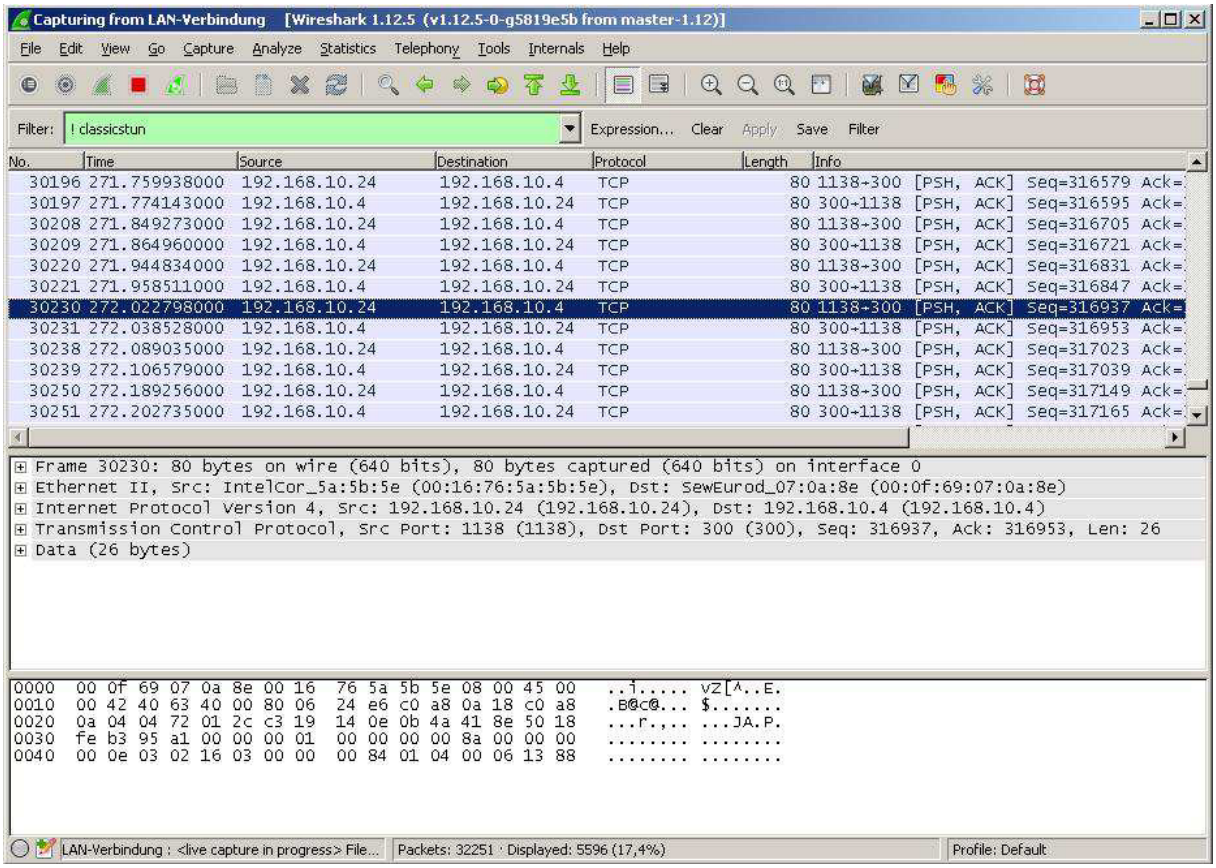

Abbildung 24: Wireshark Protokollierung des Drehzahlbefehl

Im nächsten Schritt wird eine einfache Programmierung mit LabVIEW etabliert, die dazu dienen soll, einfache hexadezimale Eingaben via TCP zu senden um so die Datenpakete von Movitools nachzubilden.

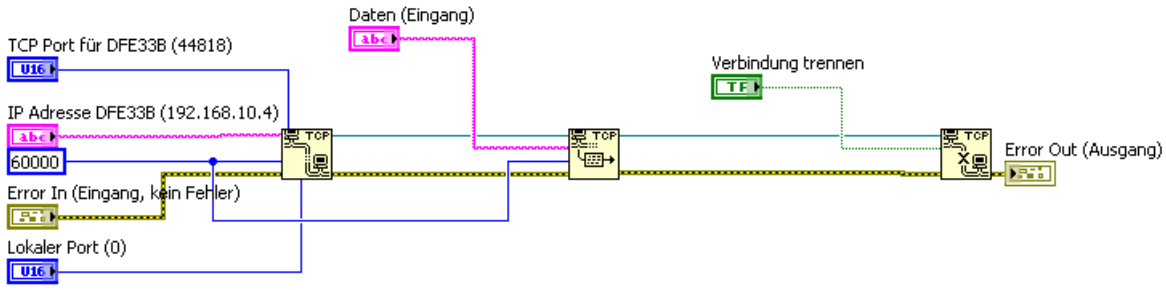

Abbildung 25: Einfache TCP sende Programmierung

Im Datenfeld auf dem Frontpanel werden die einzelnen Datenpakete, die von Movitools geschickt wurden, direkt eingegeben. Das Programm als solches besteht lediglich aus einem einfachen TCP-Verbindungs-VI, einem TCP-Sende-VI und einem TCP-Trennungs-VI. Hiermit soll versucht werden, ob es möglich ist, die gleichen Datenpakete mittels LabVIEW zu senden, wie sie auch mit dem SEW-eigenen Programm gesendet werden.

3.2 Programmierung

Die vollständige Programmierung besteht aus insgesamt sechs Abschnitten, die im Folgenden einzeln erläutert werden sollen. In den Abbildungen 38 und 39 sind die gesamte Programmierung und das dazugehörige Frontpanel dargestellt.

3.2.1 Abschnitt 1: Aufbau der TCP Verbindung

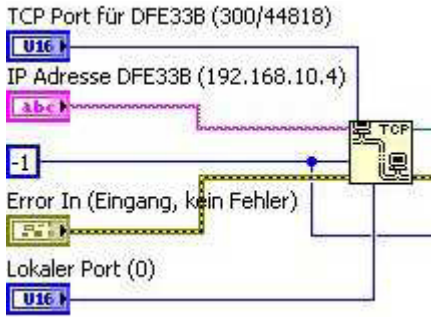

Abbildung 26: Blockdiagramm, Aufbau der TCP Verbindung

Den Anfang des Blockdiagramms bildet der Verbindungsaufbau mit dem Netzwerk. Hierzu wird ein „TCP Open Connection“-VI verwendet. In dieses werden an den Anschlussstellen die nötigen Daten wie folgt eingegeben.

- TCP Zielport
- IP Adresse des Zielgeräts
- Timeout
- Fehler Eingang
- Lokaler Port

Die Dateneingabe für den Zielport, die IP Adresse sowie den lokalen Port findet auf dem Frontpanel im oberen ersten Bereich statt.

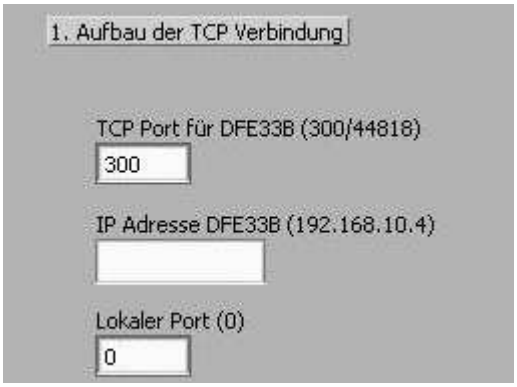

Abbildung 27: Frontpanel, Eingabe für die Verbindungsdaten

Der Zielport kann auf 300 oder 44818 gesetzt werden. Die Portnummer 300 steht hierbei für das oben beschriebene Simple Movilink Protocol, während die Portnummer 44818 für ein EtherNet/IP Protokoll steht. Laut dem Systemhandbuch von SEW sind dies, neben dem Modbusport 502, die beiden einzigen TCP Ports, auf die der Frequenzumrichter anspricht. Der lokale Port wird auf 0 gesetzt, wodurch das Betriebssystem selbständig einen Port für den Ausgangspunkt der Verbindung wählt.

Als Ziel IP-Adresse wird die IP des DFE33B eingeben, diese lautet 192.168.10.4. Der Timeoutanschluss des VIs wird mit einer Konstanten belegt, die auf -1 gesetzt wird. Dadurch

wird die Timeoutzeit unendlich. Hierdurch wird verhindert, dass die Verbindung nach einer bestimmten Zeit ohne Eingabe in den nachfolgenden Abschnitten beendet wird.

Die Errorline wird bis zum Ende des Blockdiagramms durchgezogen um mögliche Fehler in der Programmierung zu erkennen und diese im Nachhinein auslesen zu können.

3.2.2 Abschnitt 2: Handbetrieb einschalten

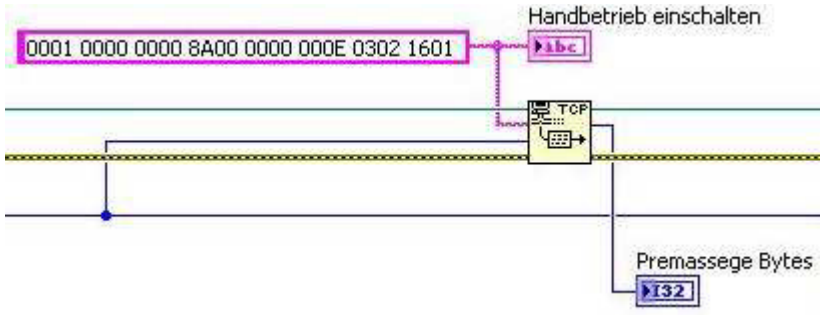

Abbildung 28: Blockdiagramm, Handbetrieb einschalten

Im zweiten Abschnitt wird ein „TCP Write“ VI verwendet, um einen fixen Datensatz zu senden (Abbildung 28), der dem Handbetrieb Starten Befehl von Movitools entspricht. Dieser Datensatz wird auch auf dem Frontpanel angezeigt und mittels des Anzeigeelements „Premassege Bytes“ kann die Länge des Datensatzes kontrolliert werden. Wenn das Programm gestartet wird muss diese Anzeige auf 16 springen.

Abbildung 29: Frontpanel, Anzeige für den Start des Handbetrieb

3.2.3 Abschnitt 3: Drehzahl Umwandlung und Bildung des Datensatz

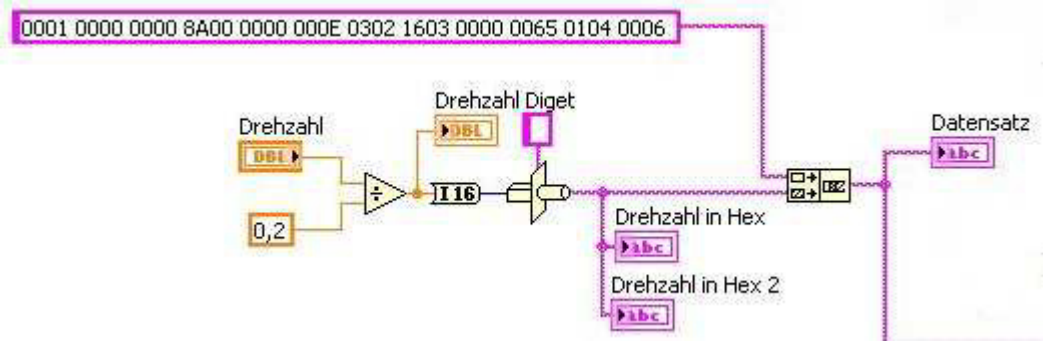

Abbildung 30: Blockdiagramm, Drehzahleingabe

In diesem Teil des Programms wird die Eingabe der Drehzahl in den Datensatz für den Frequenzumrichter implementiert. Die Drehzahl wird zunächst auf dem Frontpanel (Abbildung 31) eingegeben. Zusammen mit der Konstanten 0,2 werden diese Werte in ein Divisions-VI eingespeist. Aus der Division ergibt sich der Wert, der in dezimaler Schreibweise dem Wert der Drehzahl in hexadezimaler Schreibweise entspricht. Es ist nötig, diese Umrechnung durchzuführen, da der Motor pro Diget um 0,2 min^{-1} schneller dreht. Das Drehzahl Diget wird darauf auf dem Frontpanel angezeigt, um die Berechnung zu überprüfen. Anschließend wird

die Doublezahl in ein Integer 16 Wert umgewandelt. Doublewerte können im Gegensatz zu Integerwerten auch aus Nachkommastellen bestehen, was bei hexadezimalen Werten nicht möglich ist, daher ist die Umwandlung von Double nach Integer nötig.

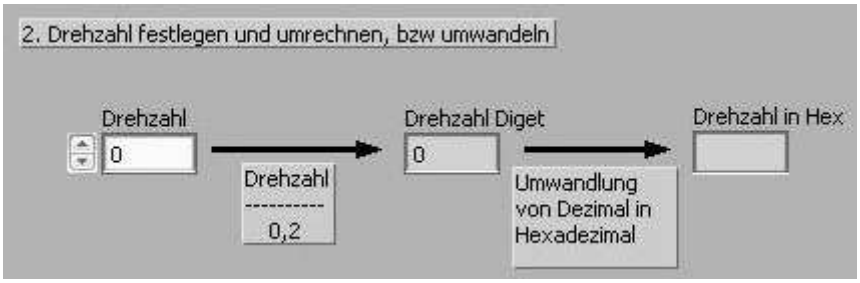

Abbildung 31: Frontpanel, Drehzahleingabe

Der Drehzahlwert wird nun durch einen Formatwandler in einen Hexwert gewandelt. Hierzu wird das gewünschte Format als leeres Datenfeld im oberen Teil eingestellt und die Anschlussklemme mit dem Ausgang des Integer-16-Wandlers verbunden. Hiernach wird der Drehzahlwert als hexadezimaler Wert abgegriffen und auf dem Frontpanel dargestellt (Abbildung 31). Nachfolgend wird die zwei Byte Drehzahl mit dem 24 Byte langen Datensatz zusammengefügt, um so das vollständige TCP Datenpaket zur Übermittlung wie durch Movitools zu bilden. Die ersten 16 Byte entsprechen hier der Fortführung des Handbetriebs, die nachfolgenden 8 Byte stehen für Daten wie die Beschleunigung des Motors, die sogenannte Rampe und andere wichtige Daten für den Betrieb des Motors.

Das vollständige TCP-Paket wird an die Zeitschleife des vierten Abschnitts weitergeleitet und auf dem Frontpanel dargestellt (Abbildung 34).

3.2.4 Abschnitt 4: Zeitschleife zum Senden der Drehzahl

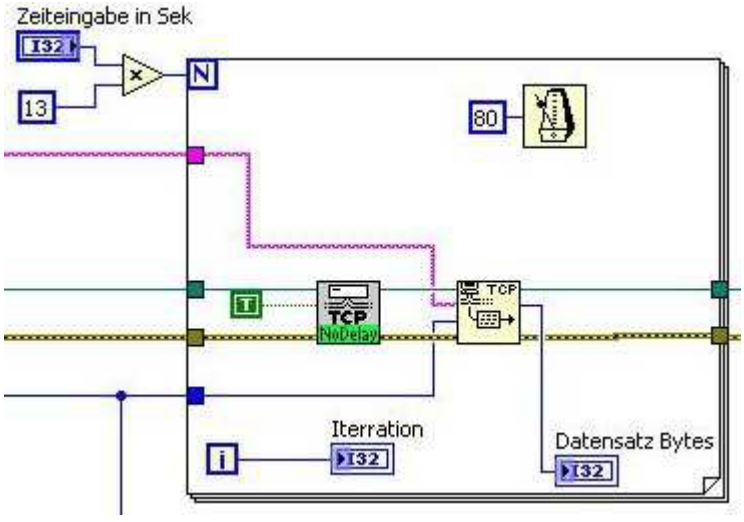

Abbildung 32: Blockdiagramm, Zeitschleife zum Senden der Drehzahl

Damit das Datenpaket zum Drehen des Motors wie auch durch Movitools alle 0,08 Sekunden gesendet wird, muss das „TCP Write“ VI in eine Schleife eingebunden sein.

Die Schleifenbedingung benötigt als Eingabe eine Iterationsanzahl. Diese wird benutzt um die Zeit festzulegen, die der Motor drehen soll. Da die Iteration im Normalfall jede Millisekunde um einen Schritt voranschreitet, muss innerhalb der Schleife eine Wartebedingung einge-

setzt werden, diese wird durch das VI mit dem Metronom Symbol erzeugt. Die Konstante, die dem VI eingegeben wird, entspricht der Zeit in Millisekunden die gewartet werden soll. Die Iterationszahl ergibt sich aus der Eingabe auf dem Frontpanel und der Multiplikation mit einer Konstante. Die Zeit, die der Motor dreht, ergibt sich somit aus der Wartezeit sowie der Iterationszahl und wird wie folgt berechnet:

$$x \cdot 13 \cdot 80\ [ms] = y\ [ms]$$

Wie zu erkennen ist, entspricht die Eingabe somit annähernd der Zeit in Sekunden, die der Motor drehen soll.

Aufgrund der geringen Sendezeit und der nur 26 Byte großen Datenpakete tritt der Nagle-Algorithmus in Kraft. Dieser muss unterbunden werden, da ansonsten ein Senden wie durch Movitools nicht möglich ist und die Pakete zusammengefasst werden und nur alle 0,2 Sekunden gesendet werden (siehe Erläuterung des Nagle-Algorithmus, Seite 13).

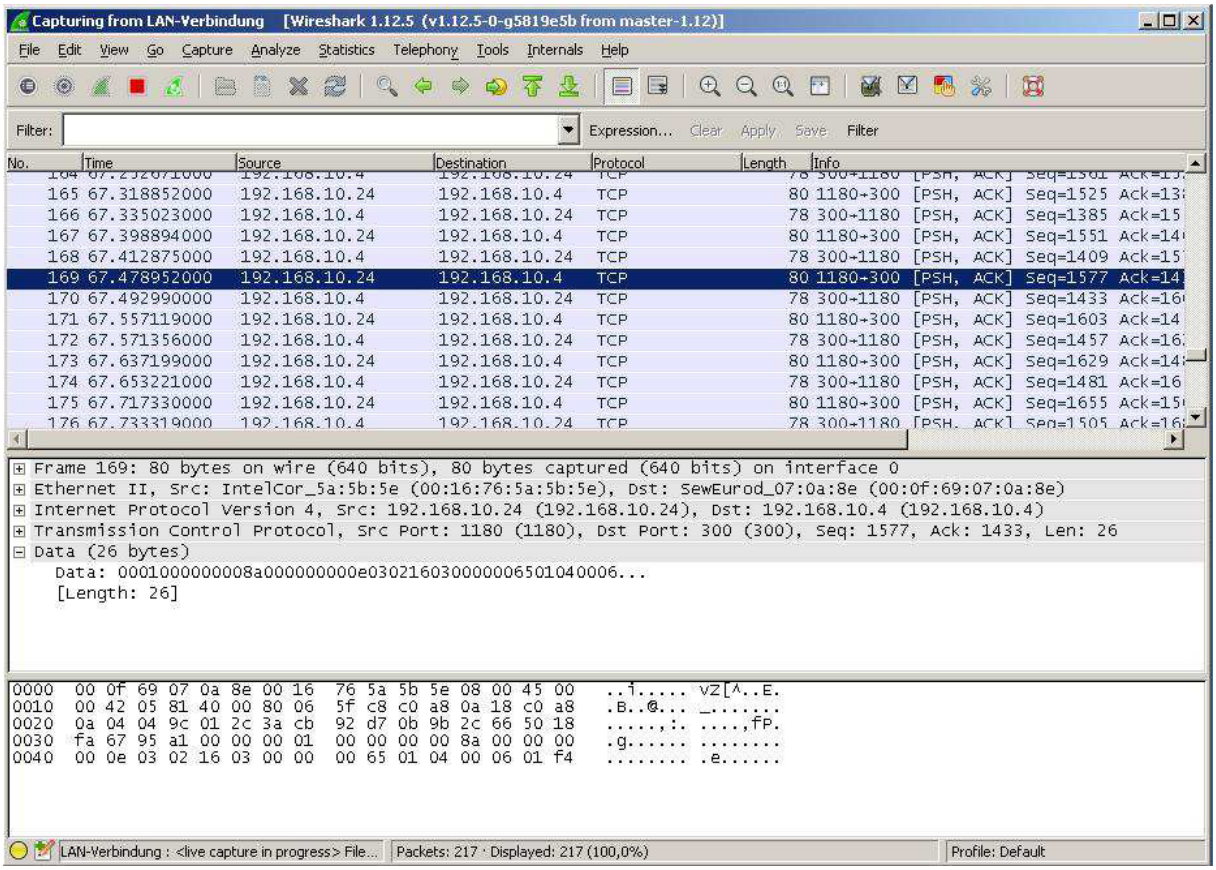

Abbildung 33: Senden der Drehzahl via LabVIEW

Das Umgehen des Nagle-Algorithmus übernimmt in diesem Fall das VI, welches dem „TCP Write“ VI vorgeschaltet ist (NoDelay). Dieses VI schaltet den Nagle-Algorithmus für diese Verbindung aus. Hierzu erhält es zum Einen die Verbindungsdaten und eine True-Konstante. Wird die Konstante Auf „False“ gesetzt, wird das VI nicht aktiviert und der Nagle-Algorithmus tritt wieder in Kraft. Die Drehzahl wird nun mit jedem Schleifendurchlauf an den Frequenzumrichter übermittelt. Zur Kontrolle der Datenlänge wird auf dem Frontpanel die Anzahl der Datenbytes ausgegeben. Die Datenpakete sollen immer eine Länge von 26 Byte haben.

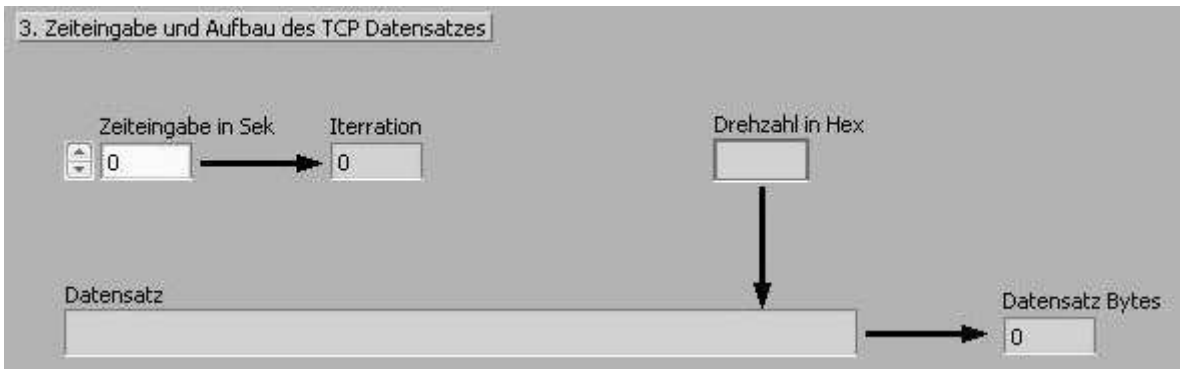

Abbildung 34: Frontpanel, Zeiteingabe

3.2.5 Abschnitt 5: Senden des Haltebefehl

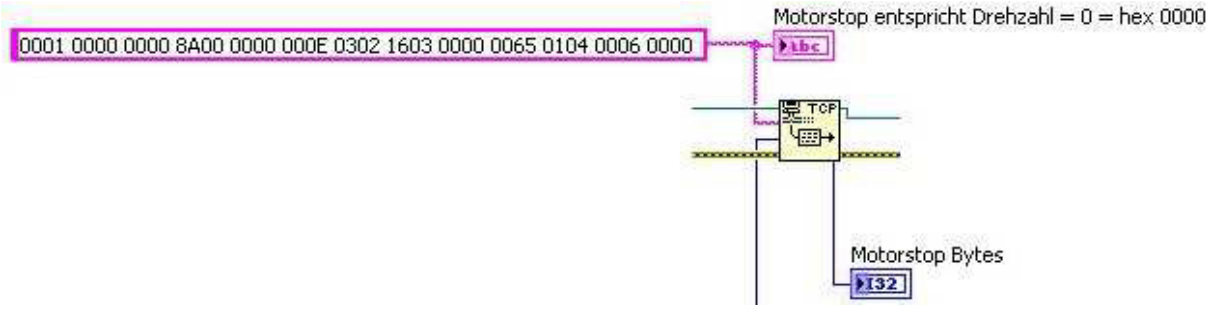

Abbildung 35: Blockdiagramm, Senden des Haltebefehls

Im vorletzten Abschnitt wird im Anschluss an die vorherige Zeitschleife der Motorstopp über ein weiteres „TCP Write“ VI ausgeführt. Hierzu wird ein Drehzahlbefehl ähnlich dem Vorherigen gesendet, jedoch mit dem Unterschied, dass hier das Datenpaket eine Konstante ist, die als Drehzahl den Wert 0 hat. Dies ist in dem TCP-Paket durch die letzten vier Nullen repräsentiert. Auch hier wird die Länge des Motorstoppbefehls und der Befehl selber auf dem Frontpanel angezeigt.

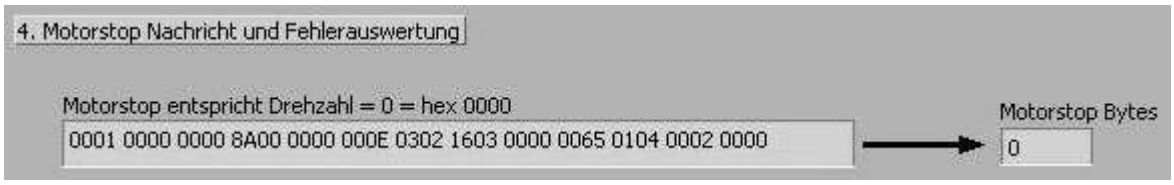

Abbildung 36: Frontpanel, Motorstopp

3.2.6 Abschnitt 6: Trennen der TCP Verbindung

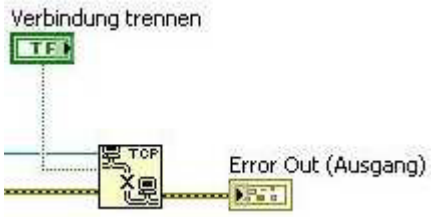

Abbildung 37: Blockdiagramm, Trennen der TCP Verbindung

Den Abschluss des Blockdiagramms bildet die Trennung der TCP Verbindung. Zu diesem Zweck wird ein „TCP Close Connection“ VI eingesetzt. Wird auf dem Frontpanel der Button zum Trennen der Verbindung betätigt schaltet der True-False Schalter die Bedingung um und die Verbindung zwischen dem Computer und dem Frequenzumrichter wird beendet.

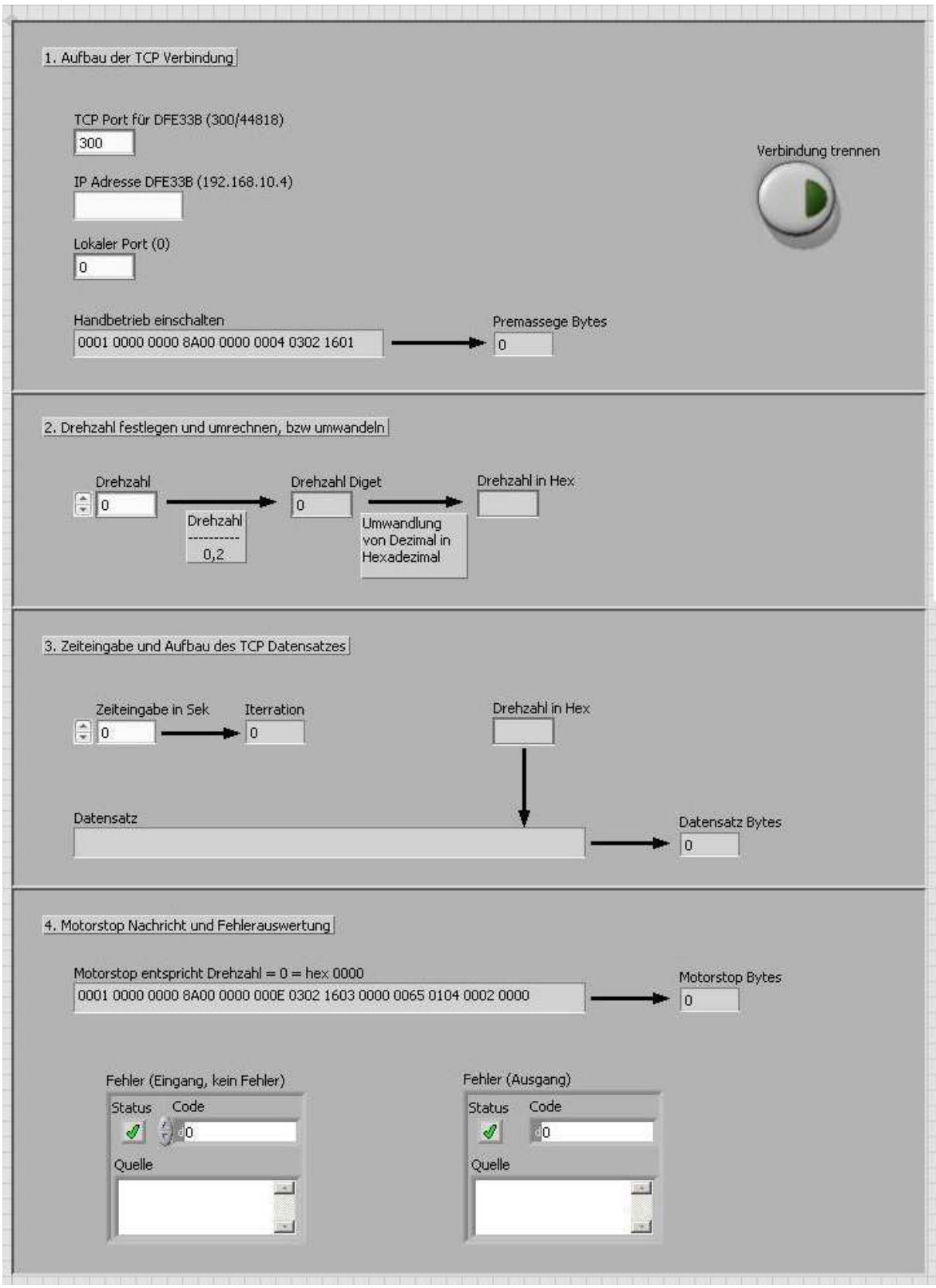

Abbildung 38: Frontpanel der Programmierung

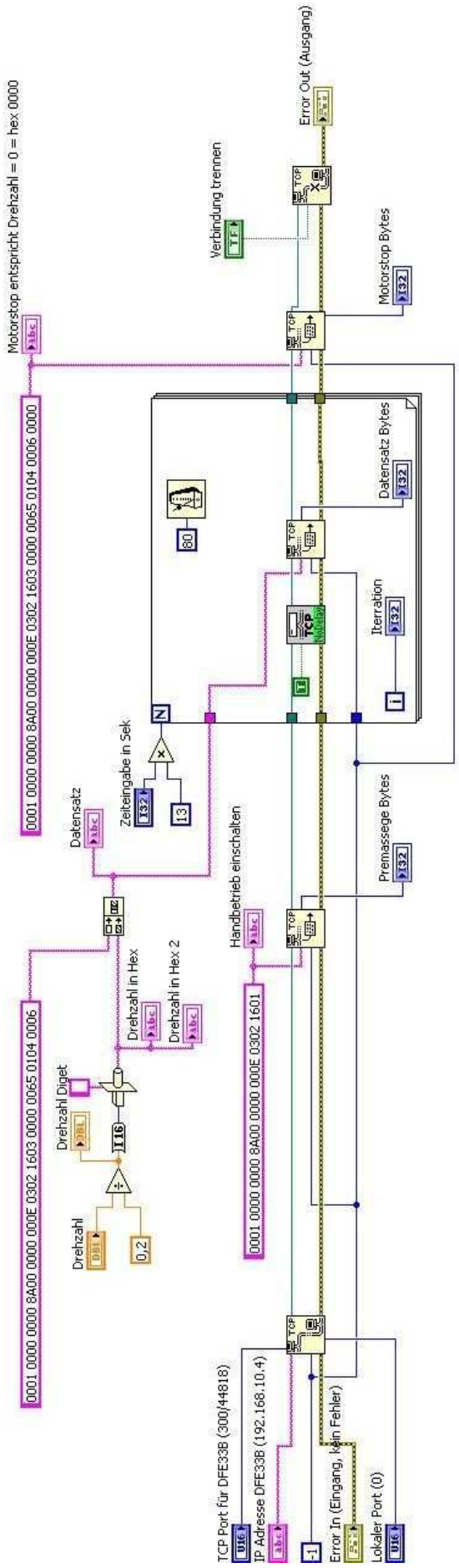

Abbildung 39: Blockdiagramm der vollständigen Programmierung

3.3 Fazit zur Programmierung

Die Programmierung sendet die Telegramme wie es vorgesehen ist, jedoch ist die Antwort vom Frequenzumrichter nicht genau lesbar und der Motor dreht sich nicht (Abbildung 40).

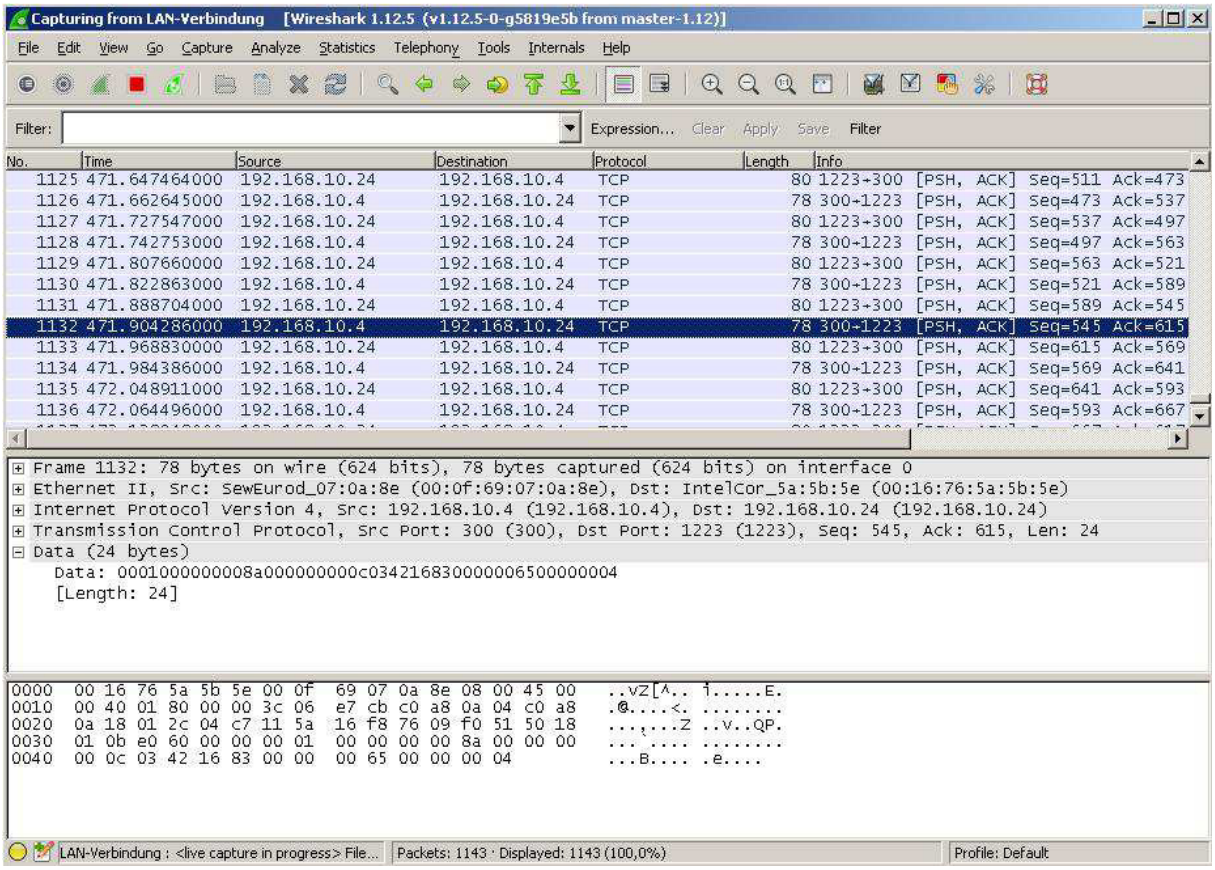

Abbildung 40: Antwort des Frequenzumrichters auf das LabVIEW Programm

Wie erkennbar ist, antwortet der Frequenzumrichter mit einem Telegramm, welches dem ihm zugesandten Telegramm ähnelt. Jedoch fehlen die letzten 2 Byte, die eigentlich die Drehzahl des Motors widerspiegeln sollten. Es ist davon auszugehen, dass dieses eingehende Datenpaket eine Fehlermeldung ist, die jedoch nicht ausgelesen werden kann. Auch der Frequenzumrichter selbst gibt auf dem Error-Display und am Handbediengerät keine Fehlermeldung aus.

Eine tiefergehende Analyse ist leider nur bis zu einem bestimmten Grad möglich, da die direkte Programmierung der VIs nicht einsehbar ist. Eine Steuerung über TCP sollte allerdings möglich sein, da der Motor auch mittels Movitools via TCP gesteuert werden kann. Die Programmierung von Movitools ist zudem ebenfalls nicht einsehbar und somit auch nicht näher analysierbar.

4. SEW eigenes Modbus Programm

SEW stellt ein LabVIEW Programm zu Verfügung, mit dem laut eigener Aussage der Frequenzumrichter mittels Modbus/TCP angesteuert werden kann. Das in Abbildung 41 dargestellte Blockdiagramm, zeigt die Programmierung und soll im Folgenden oberflächlich erläutert werden.

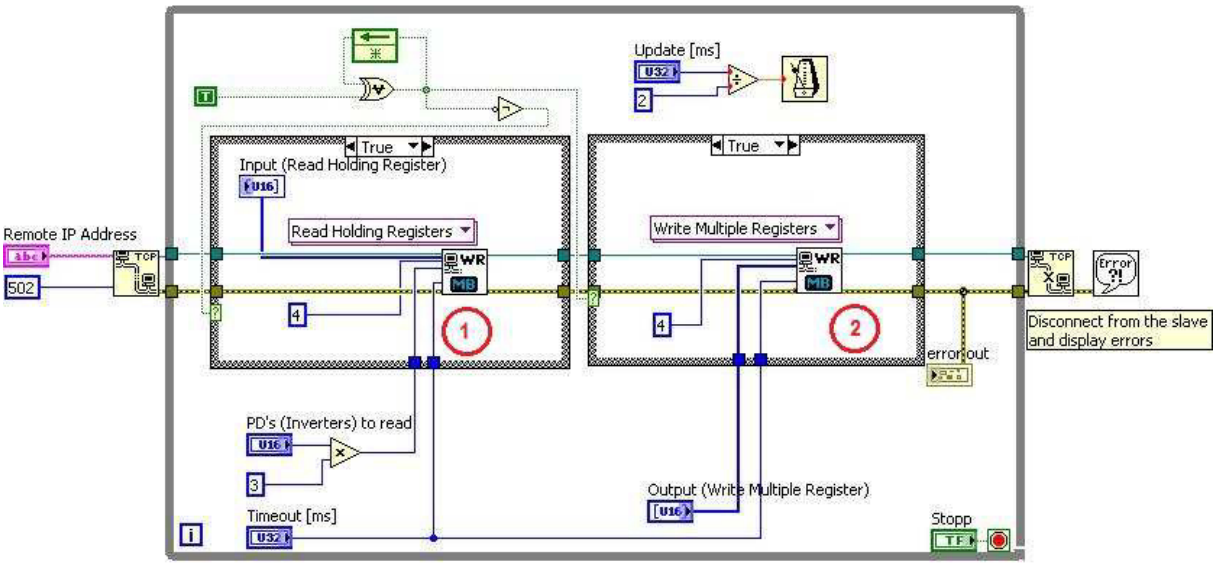

Abbildung 41: Blockdiagramm des SEW Modbus Programms

Zunächst wird durch das Programm eine TCP-Verbindung mit dem Frequenzumrichter aufgebaut und der Port 502 für Modbus Anwendungen hierfür festgelegt. Anschließend wird eine Schleife gestartet, die zum Update der Lese-und-Schreib-Befehle dient. Innerhalb der Schleife werden der Lesebefehl (Abbildung 41 1) sowie der Schreibbefehl (Abbildung 41 2) ausgeführt.

Der Schreibbefehl erhält ein Register, also eine Ansammlung an Daten vom Frontpanel und schreibt diese an die Zieladresse. Der Lesebefehl hingegen schreibt die gelesenen Daten parallel in ein Register im Frontpanel. Die Updatetime gibt die Zeit zwischen zwei Aktionen an.

Ein Test dieser Software konnte leider nicht stattfinden, da noch keine Modbuskarte vorhanden ist.

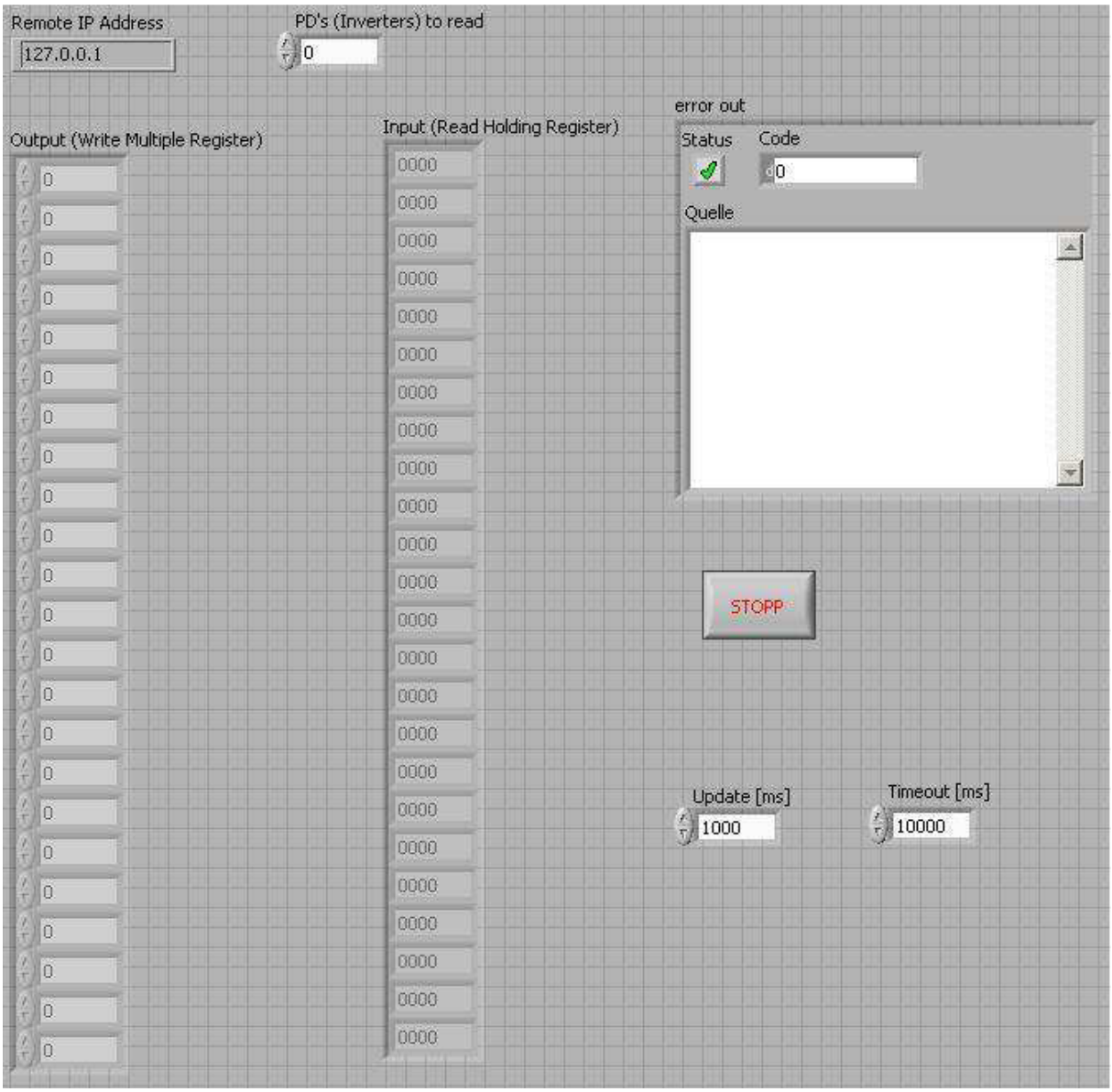

Abbildung 42: Frontpanel des SEW Modbus Programms

5. Winkelmessung

Ziel bei einer funktionsfähigen Programmierung ist es, eine Zeit sowie eine Drehzahl einzustellen und die Genauigkeit des Winkels nach Ablauf der Zeit zu überprüfen. Der Winkel berechnet sich nach:

$$\omega = 2\pi \cdot \frac{n}{60}$$

$$\alpha = \omega \cdot t = 2\pi \cdot \frac{n}{60} \cdot t$$

Um den Winkel zu überprüfen ist es vorgesehen, eine Winkelmessscheibe (Abbildung 42) zu nutzen. Hierzu wird eine Markierung an der Welle des Motors angebracht und nach Ablauf der Zeit die Änderung des Winkels mit der berechneten Winkeländerung verglichen. Hieraus lassen sich Rückschlüsse auf die Genauigkeit der Winkelstellung ziehen.

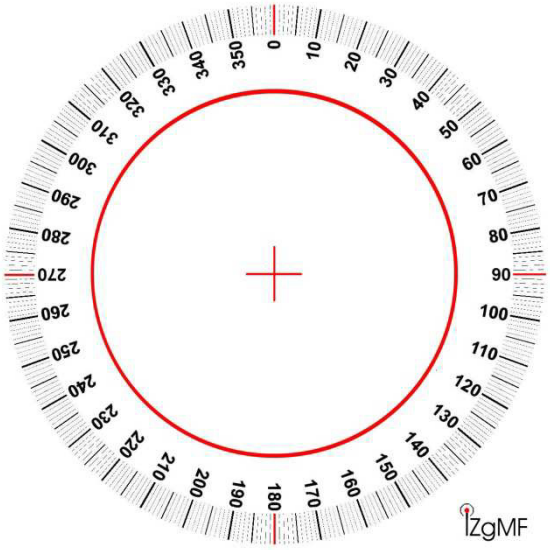

Abbildung 43: Winkelmessscheibe [17]

Eine weitere Methode ist es, die Winkeländerung über Movitools auszulesen und so einen Vergleich zuzulassen.

Im späteren Verlauf des Projekts kann zudem ein TCP Read Programm mittels LabVIEW aufgebaut werden, welches die Winkelstellung anhand eines Lesebefehls ausließt und widergibt. In Verbindung mit dem im vorangegangenen Kapitel beschriebenen Programm kann dann ein Vergleich des eingestellten Winkels und das tatsächlichen Winkels direkt auf dem Desktop stattfinden und theoretisch über eine Rückkopplung nachgesteuert werden.

6. Problematik

Wie schon erwähnt ist eines der größten Probleme, dass man die Programmstrukturen der VIs in LabVIEW nicht genau einsehen kann. Hierdurch ist eine Analyse, was die VIs tatsächlich bewirken, nicht möglich und Fehler in der Programmierung sind schwerer ausfindig zu machen.

Eine weitere Hürde war es, das der technische Support von SEW Eurodrive Hamburg fehlerhafte Informationen bereitgestellt hatte. Es hieß in einem Telefonat, dass eine Ansteuerung des DFE33B über TCP nicht möglich sei. Dies kann allerdings nicht sein, da eine Steuerung durch Movitools mittels einer EtherCat TCP Verbindung möglich ist und das Systemhandbuch ebenfalls eine Steuerbarkeit über TCP erwähnt.

Zum Start der Arbeit musste zudem die Klemme X13 (Abbildung 43) am Frequenzumrichter DFE33B überbrückt werden, um eine Steuerung mittels Movitools und TCP zu erlauben. Ist diese Brücke nicht aktiv, gibt Movitools einen Fehler aus, der jedoch im Systemhandbuch nicht direkt erläutert wird. Ein Telefonat mit dem Support ergab die nicht aktive Brücke. Nach dem Setzen dieser war eine Steuerung problemlos möglich

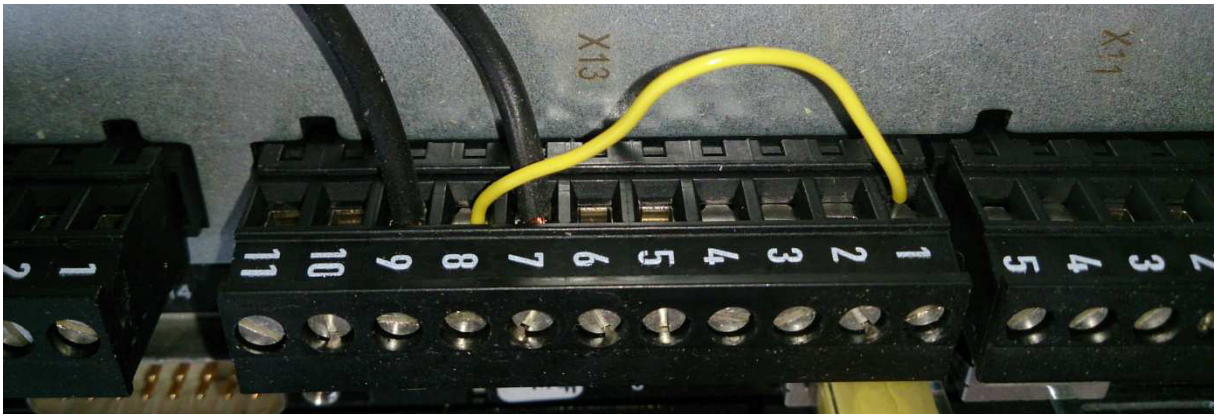

Abbildung 44: Überbrückung der Anschlussstelle X13 des DFE33B

Des Weiteren ist darauf zu achten, dass die IP-Adresse des Rechners nicht geändert wird, da dies zu Problemen bei der Kommunikation mit dem Frequenzumrichter führen kann und, wenn die Änderung nicht dokumentiert wird in eine längere Fehlersuche ausarten.

Beim Betrieb des Motors über Movitools sollte zudem in Betracht gezogen werden, dass man das Handbediengerät vom Frequenzumrichter abkoppelt, da in einigen Fällen durch Eingaben am Rechner das Handbediengerät Fehlermeldungen ausgab. War das Gerät jedoch nicht angeschlossen, traten diese Fehler nicht weiter auf. Es ist davon auszugehen, dass hier eine Rückkopplung zwischen dem Rechner und der manuellen Eingabe am Frequenzumrichter stattfindet, da beide gleichberechtigt agieren können.

7. Aussichten

Für das weitere Vorgehen im LaOla Projekt ist es zu empfehlen, die Firmen SEW und National Instuments direkt einzubinden. Beide Firmen haben Ableger in Hamburg und können so mit ihrer Erfahrung und Anwesenheit im Projekt Unterstützung leisten. Durch eine Einbindung der Firmen in das Forschungsprojekt könnte es möglich sein, Probleme, wie die, die hier zum Misserfolg der Programmierung oder wie in der Hausarbeit CAN-Bus-Ansteuerung für einen elektrischen Wellenkanalantrieb, zu vermeiden.

Ein weiteres Projekt könnte die Ansteuerung des Motors über Modbus/TCP sein, da SEW hierzu bereits, wie in Kapitel 5 erwähnt, ein eigenes Programm anbietet und somit Hilfestellung bieten könnt. Die Anschaffung einer Modbus Karte sollte hierzu überdacht werden.

Des Weiteren muss das Wellenpaddel, welches zum Erzeugen der Wellen genutzt wird am Boden des Wellenkanals befestigt werden. Diese Befestigung könnte im Rahmen einer Hausarbeit erarbeitet werden. Außerdem muss ein Wellenabsorber am Ende des Kanals installiert werden, um ein Zurückschlagen der Wellen von der Wand des Kanals zu verhindern.

Die Einbindung in den Ablauf des Bachelor Projekts könnte zudem dafür sorgen, mehr Aufmerksamkeit auf das LaOla Projekt zu lenken und kleinere Aufgabestellungen zu bearbeiten, die nicht Teil einer Hausarbeit oder einer Bachelorthesis sein können.

Quellenverzeichnis

[1] *Der Weg zur Energie der Zukunft* www.bundesregierung.de S. 1 Nr 4

[2] Artikel der Süddeutschen Zeitung. *Kabinett beschließt Atomausstieg bis 2022* 6. Juni 2011 http://www.sueddeutsche.de

[3] *Erneuerbare-Energien-Gesetz* EEG §1 Abs. 2

[4] *Erneuerbare-Energien-Gesetz* EEG §3 Nr.1, Nr.2 & Nr.3

[5] BDEW Bundesverband der Energie- und Wasserwirtschaft *http://www.bundesregierung.de/Content/DE/Artikel/2014/01/2014-01-13-bdew-energiebilanz-2013.html* (Aufgerufen im Juli 2015)

[6] Pelamis Wave Power *http://www.emec.org.uk/about-us/wave-clients/pelamis-wave-power/* (Aufgerufen im Juli 2015)

[7] Mathias Simon, Antje Klemichen. *Aufbau und Implementierung der Messtechnik in einem Wellenkanal* S. 12

[8] *Wireshark User Guide*

[9] Info Management Systems AG IMS. *SH_Checkup_NetzwerkFremdeinfluss.pdf* S. 2 http://www.ims-info.ch/support/pdf/SH_Checkup_NetzwerkFremdeinfluss.pdf (Aufgerufen Juli 2015)

[10] Elektronik Kompendium beruhend auf Patrick Schnabel *Netzwerktechnik-Fibel http://www.elektronik-kompendium.de/sites/net/0812271.htm* (Aufgerufen Juni 2015)

[11] Kevin Washburn, Jim Evans. *TCP/IP Aufbau und Betrieb eines TCP/IP-Netzes 2.Auflage* S. 281

[12] DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE. *enetipcommondescription.pdf*

[13] Camille Bauer AG. *Modbus_Grundlagen.pdf* http://www.camillebauer.com

[14] Hilscher CIFX 50-RE *http://www.hilscher.com/de/produkte/produktgruppen/pc-karten/pci/cifx-50-reomb/* (Aufgerufen Juli 2015)

[15] Erstellt nach *MODBUS over Serial Line Specification and Implementation Guide* (Modbus_over_serial_line_V1_02.pdf) www.modbus.org

[16] *LabVIEW Hilfe Datei*

[17] http://www.izgmf.de/izgmf-winkelscheibe.jpg (Aufgerufen August 2015)

[18] Hochschule Mittweida telecom group *https://www.telecom.hs-mittweida.de/teachware0.html* wireshark.pdf S.2

[19] SEW Eurodrive *Systemhandbuch Movidrive MDX60B/61B* S. 107 Ausgabe 09/2010